

**DOCENTEN
BOEK
COMPUTERKUNDE**

ICWC

DOCENTEN
BOEK
COMPUTERKUNDE

y.20BA-2
(ex2)

DOCENTEN BOEK COMPUTERKUNDE



Instituut Ontwikkeling Wiskunde Onderwijs



© 1973 Instituut voor Ontwikkeling van het Wiskunde
Onderwijs

Niets uit deze uitgave mag worden verveelvoudigd en/of
openbaar gemaakt door middel van druk, fotokopie, micro-
film of op welke andere wijze ook zonder voorafgaande
schriftelijke toestemming van de houder van het copyright.

in h o u d

WOORD VOORAF	blz.	7
HOOFDSTUK 1		
Doelstellingen Computerkunde		
1.1	Computerkunde bij het A.V.O. Prof.Dr. A. van der Sluis	" 8
1.2	Voortgezette computerkunde	" 12
1.3	Internationale aanbeveling	" 15
HOOFDSTUK 2		
Aanwijzingen voor de les		
2.1	De leerstof	" 18
2.2	Tijdschema	" 36
2.3	Opgaven	" 37
2.4	Enkele programma's	" 47
HOOFDSTUK 3		
Het "ECOL-rapport"		
I	Taaldefinitie, vertaling, executie	" 51
II	Globale vorm van ECOL programma's	" 52
III	Contextregels en compileraktie	" 53
IV	Executie	" 56
V	SUB en FUN	" 66
VI	De ECOL syntax	" 72
HOOFDSTUK 4		
Wetenswaardigheden		
4.1	Literatuurbespreking	" 76
4.2	Terminalgebruik bij de computerkunde	" 80
HOOFDSTUK 5		
Achtergrondinformatie		
5.1	Computerhardware	" 82
5.2	Datastructuren	" 89
5.3	Iets over software	" 92
5.4	Recursiviteit	" 97
HOOFDSTUK 6		
Programmaverwerking		
6.1	De ECOL schrapkaart	" 102
6.2	De ALGOL schrapkaart	" 104
6.3	Verzenden van schrapkaarten	" 105
6.4	Kosten programmaverwerking	" 106
6.5	Bestanden	" 108

Woord vooraf

HET EXPERIMENT COMPUTERKUNDE

Sinds 1968 organiseert de Commissie Modernisering Leerplan Wiskunde (CMLW) een experiment in computerkunde. Aanleiding hiertoe was het rapport nummer 3 van deze commissie. Het experiment beoogt de wenselijkheid en mogelijkheid van algemene invoering van deze leerstof te onderzoeken. Deze invoering heeft echter omvangrijke gevolgen ten opzichte van de heroriëntering van docenten en van de organisatie van het gebruik van computers voor het onderwijs. Daarom wordt voorlopig de voorkeur gegeven aan het telken jaren uitbreiden van het experiment en zo toe te groeien naar computerkunde bij het gehele voortgezet onderwijs.

HET DOCENTENBOEK

vanaf 1970 is ten behoeve van de aan het experiment deelnemende docenten een tijdschrift 'Computerkunde IOWO' uitgegeven. Dit docentenboek is ontstaan als bloemlezing uit voorgaande jaargangen van dit tijdschrift gevoegd bij gedeelten van heroriënteringscursussen. De wijze van samenstellen heeft de eenheid van stijl niet bevorderd, maar dat zal minder storen naarmate U het boek meer als naslagwerk beschouwd.

LEERBOEKEN

Uit het voorgaande volgt dat veel van de inhoud van dit docentenboek direct betrekking heeft op het eerder genoemde experiment. Zo zullen de aanwijzingen voor de les aansluiten op de in het experiment gebruikte leerboekjes, te weten:

'Computerkunde voor het A.V.O.' , deel 1 en 2

C.A.Ch.Görts, S.G.v.d.Meulen, A.v.d.Sluis, J.R.Zweerus
uitgave Wolters-Noordhof

'Werkschrift Computerkunde'

G.A.Vonk uitgave IOWO

In de tekst wordt hiernaar verwezen met de afkortingen 'Compk' en 'Wrks'. Gebruikers van andere methoden zullen met de lesaanwijzingen zeker ook hun voordeel kunnen doen. In het bijzonder zullen de aanwijzingen voor de programmaverwerking ook van toepassing zijn voor de gebruikers van

'Elementaire Computerkunde voor mavo en havo'

W.van den Camp, H.B.Emanuel, J.B.Lubke, P.Th.J.Ploeger
uitgave Meulenhoff Educatief

EDITIE 1971

Bijdragen voor een volgende druk van het docentenboek worden met belangstelling tegemoet gezien, evenals op- of aanmerkingen op deze aflevering. Het adres is dat van het 'Instituut Ontwikkeling Wiskunde-Onderwijs' (IOWO), de uitvoerende instelling van de commissie genoemd in de eerste alinea. Ik dank hierbij al degenen die hun pennevruchten voor dit docentenboek hebben afgestaan.

november 1971

G.A.Vonk
IOWO
Tiberdreef 4
Utrecht

BIJ DE TWEEDE DRUK

Deze editie is globaal gelijk aan de vorige. Gewijzigde omstandigheden hebben enkele aanpassingen noodzakelijk gemaakt, o.a. uitbreiding van §2.1 en het vervallen van de oorspronkelijke paragrafen 4.2, 4.4 en 4.5.

juli 1973

B.H.Knip, G.A.Vonk

1 Doelstellingen computerkunde

1.1 COMPUTERKUNDE BIJ HET ALGEMEEN VOORTGEZET ONDERWIJS

Het experiment computerkunde mag zich inmiddels verheugen op deelname van het beroepsonderwijs. Voor deze vorm van onderwijs is onderstaand artikel niet bedoeld. Zo zal bijvoorbeeld niet altijd een wiskundeleraar aanwezig zijn om het onderwijzen van computerkunde tot zijn taak te rekenen. Aan het beroepsonderwijs nu om te bepalen in hoeverre zij van onderstaande beweegredenen moeten afwijken. Het artikel van Prof. Dr. A. van der Sluis is overgenomen uit 'Euclides', jaargang 46 nummer 3.

Inleiding

Dit stuk beoogt uiteen te zetten dat computerkunde een noodzakelijk onderwerp is op elk leerplan voor algemeen voortgezet onderwijs (verder af te korten als AVO, waaronder begrepen VWO), dat een inleiding tot dit vak in het wiskunde-programma thuishoort, en door de wiskundeleraar gegeven moet worden. Een leerplanschets wordt gegeven. Men kan dit stuk zien als een uitwerking van [1], echter geheel voor verantwoordelijkheid van de auteur. Het gaat hier dus duidelijk niet om beroepsonderwijs. Dit betekent niet dat op scholen voor beroepsonderwijs computerkunde niet net zo nodig zou zijn. Maar de doeleinden en middelen kunnen daar wel anders zijn.

1 Doel van het AVO

Een groot aantal onderwerpen komt op zichzelf genomen in aanmerking om in het schoolprogramma opgenomen te worden. Het is daarom duidelijk dat men zeer scherpe criteria zal moeten aanleggen alvorens een nieuw onderwerp op te nemen, temeer waar dit over het algemeen zal impliceren dat iets anders moet wijken. Overigens dienen dezelfde criteria te worden aangelegd om te bezien of dat wat is, kan blijven.

Deze criteria hangen uiteraard samen met de doelstellingen van het onderwijs als geheel. Aangezien het, voor zover ik heb kunnen nagaan, al weer 10 jaar geleden is dat in dit tijdschrift aandacht gegeven is aan de algemene doelstellingen van het onderwijs (n.l. in het artikel van W. Peremans, *Euclides* 36 (1960), p. 129-164), lijkt het me verantwoord hierop allereerst in te gaan.¹) Temeer waar er inmiddels een mammoetwet in werking is getreden waarin niets over deze materie te vinden is, en er in het voorjaar 1968 voorstellen leerplan rijks-scholen zijn verschenen waar het wiskundeleerplan een van de weinige is die niet met een doelstelling ingeleid worden. Het is kennelijk een heet hangijzer, begrijpelijk overigens gezien de snelheid waarmee onze maatschappij en maatschappijbeschouwing evolueren, en daarmee de inzichten betreffende het doel van het onderwijs. Desondanks: hoe kan men leerplannen opstellen zonder een welomschreven doel?

Zeer schematisch gezegd zie ik als belangrijkste taken van het AVO de leerlingen voor te bereiden op

- (a) *het leven in de wereld van morgen*
- (b) *het werken in de wereld van morgen*
- (c) *verdere studie*

Aan elk van deze taken is inherent persoonlijkheidsvorming van de leerling, waaronder begrepen zaken als het bijbrengen van verantwoordelijkheidsgevoel, zin voor samenwerking e.d. Aan dit aspect ga ik in deze beschouwing voorbij, o.a. omdat het me voorkomt dat dit minder met het leerplan dan met de wijze van onderwijs geven samenhangt.

Over elk van deze taken een paar woorden:

Taak (a). De leerling moet leren hoe een samenleving functioneert, hoe deze zich ontwikkelt, welke krachten daarin een rol spelen, en hoe deze krachten beheerst kunnen worden.

Taak (b). Voor elke taak zijn zekere specifieke kennis en bekwaamheid vereist. Het is niet de taak van het AVO deze bij te brengen, maar wel moet het er de basis voor leggen. Maar ook en vooral moet het algemene bekwaamheden en niet-specifieke kennis bijbrengen. Onder algemene bekwaamheden versta ik bekwaamheden als: tot de kern van een zaak door te dringen, te leren, je kennis te gebruiken, een werk af te ronden, te organiseren. Nu specifieke kennis en bekwaamheid aan een zo snelle veroudering onderhevig zijn wordt de beteke-

nis van deze algemene bekwaamheden steeds groter. De hoedanigheid van een vak, die de ontwikkeling van deze algemene bekwaamheden bevordert, behoort tot de vormende waarde ervan, een vaag begrip waarmee men gewoonlijk de mate aanduidt waarin dit vak in staat is een wenselijke habitus aan te kweken, iemand te leren om andere dingen die met het betrokken vak in feite niets uitstaande hebben, beter te doen. Een vak dat traditioneel hoog om zijn vormende waarde wordt aangeslagen is de wiskunde.

Taak (c). In principe verschilt deze taak niet van de vorige, maar in zijn implicaties wel degelijk, doordat de verdere studie wel heel specifieke eisen kan stellen (men denke aan latijn en wiskunde als noodzakelijke voorbereiding voor tal van studierichtingen), waaraan zonder de nodige aandacht op school niet voldaan kan worden, en ook diverse algemene bekwaamheden krijgen hier speciale nadruk.

Het spreekt vanzelf dat deze taken evenzovele criteria vormen die men dient te hanteren bij het overwegen of een onderwerp een plaats op het leerplan verdient of niet. In verband met het grote belang ervan licht ik echter het onderdeel vormende waarde uit (b), en maak hiervan een apart criterium, criterium (d). Het belang van onderwijs in een bepaald vak is een soort gewogen som van de maten waarin het aan al deze criteria voldoet. Hoe moeilijk deze dingen ook te wegen zijn, het is duidelijk dat opstellers van schoolprogramma's de plicht hebben te streven naar optimalisatie van de output. Dit geldt voor elk te onderwijzen vak apart, maar ook voor het onderwijspakket als geheel, vooral wanneer, zoals ten onzent, de leerling althans in de eerste 3 of 4 jaren bij het AVO slechts de keus heeft uit een paar standaardpakketten.

Heeft men zich er bij de opstelling van het leerplan voor enig vak, bijv. voor de wiskunde, inderdaad bewust rekenschap van gegeven in welke mate elk onderdeel ervan de bovengenoemde doeleinden dient, en in welke mate dezelfde doeleinden met minder moeite bereikt zouden kunnen worden door een andere (zij het wellicht minder bij de traditie of de universitaire mode aansluitende) keuze van de stof?

2 Het belang van computerkunde

In het licht van de voorafgaande beschouwingen willen we nu het vak computerkunde bezien. We zullen ons hierbij ter wille van lengte en overzichtelijkheid van dit artikel zeer moeten beperken; elk criterium is op zich goed voor een heel artikel.

Criterium (a). Het patroon van onze samenleving ontwikkelt zich in snel tempo en de computer is hierin een belangrijk instrument. Reeds nu heeft de computer vele belangrijke veranderingen teweeg gebracht, terwijl hij nog pas 10 jaar op enige schaal in gebruik is. Slechts zeer weinigen realiseren zich (of moet ik zeggen: hebben genoeg fantasie) in welke mate de computer in de toekomst ons leven zal gaan beïnvloeden. We kunnen ons aan deze kracht niet onttrekken, en moeten er ook niet bang voor zijn, maar hem accepteren en trachten ermee vertrouwd te raken, juist als met andere belangrijke krachten in de samenleving. De computer brengt een sociale en culturele revolutie teweeg, waarvan het belang vergeleken kan worden met dat van de industriële revolutie in de vorige eeuw. Werd toen routinematige *lichamelijke* arbeid door machines overgenomen, nu is dit het geval met routinematige *geestelijke* arbeid. Het feit dat veel activiteiten die altijd van intellectuele aard geacht werden nu routinewerkzaamheden blijken te zijn, die men kan programmeren voor een computer, is ook conceptueel een gebeurtenis van de eerste orde, die een nieuw licht werpt op menselijke activiteiten.

Het huidige gebrek aan inzicht bij de doorsnee burger omtrent de wijze waarop deze nieuwe kracht in onze samenleving functioneert wekt verwarring en onzekerheid. Maar ook bevordert dit dat deze kracht ons gaat overheersen in plaats van andersom. In dit verband mag nog eens aan Norbert Wiener's bekende

¹ Na het gereed komen van dit artikel verscheen juist het artikel 'De doelstellingen van het wiskundeonderwijs' van H. van der Hak (*Eucl.* 45 (1970), p. 289-299). Het is niet de bedoeling van het huidige artikel met de heer van der Hak in discussie te treden (auteur)

Wij wijzen ook op 'Naar een nieuw onderwijsprogramma voor de wiskunde' van J. van Dormolen (*Eucl.* 46 (1970), p. 1-7) (redactie).

reactie herinnerd worden op de vraag of het gevaar dat robots ons zouden gaan overheersen niet tamelijk denkbeeldig is omdat men dan toch eenvoudig de stekker uit het stopcontact kan trekken. Wiener's antwoord was dat dit middel wel eens erger dan de kwaal zou kunnen zijn; immers, naar mate men meer aan de robots overlaat, wordt de samenleving meer ontwricht wanneer men ze buiten gebruik stelt. Computers in de hand te houden zal een belangrijke taak zijn van deze generatie, en moet niet worden overgelaten aan een kleine clan van computermagiers.

Criterium (b). Wat onder (a) gezegd werd over de revolutie die computers te weeg brengen raakt natuurlijk ook en heel speciaal de werksituatie.

Veel vroeger door mensen vervulde functies worden nu verricht door computers. Een computer kan sneller en vollediger alle mogelijkheden overzien om een machine of wagenpark te benutten dan mensen. Een computer kan de toestand van een bedrijf op de huidige minuut weergeven. Aan te houden voorraden kunnen veel nauwkeuriger beheerd worden. Bibliografische gegevens kunnen veel sneller teruggevonden worden. Maar ook kan de computer allerlei dingen doen die vroeger helemaal niet gebeurden, zoals het door simulatie schatten van het effect van maatregelen die men overweegt te nemen.

De taken die door mensen moeten worden vervuld verschuiven hierdoor uiteraard aanzienlijk. Wegens de universele bruikbaarheid van de computer kan men rustig zeggen dat een groot percentage van de huidige scholieren in hun latere werkkring op een of andere manier met computers te maken krijgt. Zij zullen hun werk verrichten door middel van computers, (ook al zullen ze hem misschien niet zelf programmeren of bedienen) en zij zullen werk uitvoeren dat geheel of gedeeltelijk door computers is voorbereid. Het is daarbij niet van belang dat zij precies weten hoe men de meest uiteenlopende dingen met een computer doet. Het gaat erom dat zij er beseft van hebben wat een computer zoal kan doen, zodat zij in voorkomende gevallen naar een computerexpert kunnen gaan.

Dit beseft, niet de vakbekwaamheid is wat alle leerlingen op school moeten opdoen.

Criterium (c). Aangezien in zoveel beroepen de computer een belangrijke rol gaat spelen is het duidelijk dat vele leerlingen de computer bij hun verdere opleiding zullen tegenkomen. Dit is zeker ook waar voor studie aan de universiteit.

Reeds in de nabije toekomst zullen er slechts weinig studierichtingen zijn waarin men niet in meerdere of mindere mate met de computer in aanraking komt. Het zijn bepaald niet alleen de studenten in de exacte wetenschappen die tijdens hun studie met de computer moeten werken, maar ook de studenten in vakken als rechten, humaniora en literaire wetenschappen. In een officieel rapport uit de USA (zie [2]) schat men dat van deze laatste categorieën omstreeks de helft enig computerwerk zullen doen.

Criterium (d). De vormende waarde van het leren werken met een computer kan nauwelijks overschat worden. Het kweekt n.l. een sterk operationele, algoritmische en organisatorische instelling aan. Met deze termen bedoel ik het volgende.

Operationeel. Een operationele instelling is het streven kennis en informatie die tot je komt (en die waarover je reeds beschikt) operationeel te maken, d.w.z. geschikt om mee te werken. Dit betekent een evalueren ervan, en je bewust maken wat het impliceert, wat je er aan hebt. De leerlingen blijken vaak wel allerlei dingen te weten, maar het schort steeds weer aan deze operationaliteit. Een gebrek hieraan trachten ze dan vaak voor zichzelf en hun leraar te verbergen achter breedspakigheid, en vaak schoppen ze het hiermee een heel eind. Bij een computer komen ze echter met breedspakigheid niet ver. Dan moeten ze zich precies realiseren wat ze weten, wat er gegeven is, en hoe ze hiermee moeten handelen om een gesteld doel te bereiken.

Algoritmisch. Het woord algoritme betekent zoiets als een rekenwijze. Algemener verstaan we onder algoritme elk eenduidig voorschrift om een proces (bijv. een rekenproces of een breiproces of een telefoonproces) uit te voeren met eindig veel welbegrepen en uitvoerbare handelingen. Algoritmiek is dan de discipline om algoritmen te herkennen, te construeren en te analyseren. Deze algoritmen moeten niet alleen tot stand brengen wat van hen verlangd wordt, maar moeten gewoonlijk ook nog aan andere condities voldoen zoals eenvoud en efficiency. Een algoritmische instelling is het streven de routineaspecten in allerlei processen te ontwaren, algoritmen voor gegeven processen te ontwikkelen.

Het belang hiervan is duidelijk. Processen die creativiteit vereisen kunnen slechts worden uitgevoerd door enkelingen, routineprocessen kunnen worden uitgevoerd door velen. En deze processen kunnen dan als bouwstenen gebruikt worden voor meer gecompliceerde processen. Dit mechanisme, creativiteit om te zetten in routine, ligt ook ten grondslag aan veel wetenschappelijke vooruitgang. Wat gisteren nog moeilijk was en veel hoofdbrekens kostte is vandaag routine geworden zodat men zich kan concentreren op verder reikende problemen. Er zijn tal van voorbeelden dat de ontwikkeling van de wetenschap

langdurig stagneerde doordat allerlei kennis die al wel vergaard was nog niet tot een goed bruikbaar apparaat was omgesmeed (men denke zich maar eens in dat wij nog zo moesten rekenen zoals de Romeinen het deden of dat wij moesten integreren zoals Archimedes). Een mens is nu eenmaal niet in staat om veel problemen tegelijkertijd te behandelen; hij moet ze inkapselen, hij moet verdelen om te kunnen heersen. Dit is ook van belang in de school. Er is tegenwoordig een verschuiving van routine naar wat begrip wordt genoemd. Routine is haast een smerig woord geworden. En inderdaad is er in het verleden veel onbruikbare routine en te weinig bruikbaar begrip aan de leerling gepresenteerd. Routine moet functioneel zijn, het moet de apparatuur zijn waar de leerlingen mee werken en dan is het onmisbaar.

Routine te vinden, te isoleren en als algoritme te formuleren is een creatieve activiteit. Een algoritmische instelling is hiervoor een vereiste.

Organisatorisch. Na het voorafgaande kunnen we hierover kort zijn. Werken met een computer betekent organiseren. Bij organiseren is het belangrijk niveau's van complexiteit te onderscheiden, alle mogelijke gevallen te voorzien en schijnbaar verschillende zaken onder één noemer te brengen. Dit is precies wat werken met de computer aankleeft.

3 Wat voor computerkundeonderwijs

Veel onderwijs vaart onder valse vlag. Iedereen beaamt dat wiskunde een belangrijk vak is en is bereid dit zo nodig met een aantal voorbeelden te staven. Het is helemaal niet moeilijk om voor de wiskunde een soortgelijke beschouwing te houden als hierboven onder 2 gedaan is voor computerkunde. Desondanks zat (en zit?) in het wiskundeprogramma veel dat niet aan de in 1 genoemde criteria voldoet.

Evenzo kan men ook zaken onder de vlag van belangrijkheid van de computerkunde laten meevaren waarop het onder 2 gestelde niet of veel minder van toepassing is. Velen zijn van mening dat bij computerkunde het binaire stelsel, flipflops en magneetringetjes belangrijke onderwerpen zijn, en sommigen menen dit zelfs ook ten aanzien van het bedrijfssysteem van de computer (dat de samenwerking tussen de diverse componenten regelt), datarepresentatie en -opslag, compilers etc.

Ik wil hier duidelijk stellen dat onderwijs in het binaire stelsel, de computerhardware en de computersoftware niet vallen onder de belangrijkheid die onder 2 voor computerkunde werd geclaimd. Deze onderwerpen zijn van tweede importantie, bezien in het kader van het onderwijs als geheel. Wat wel belangrijk is is algoritmiek (zie 2). Het gaat er om dat de leerling in staat is het principe van grote computertoepassingen te doorgronden en (kleine) eigen probleempjes m.b.v. een computer op te lossen.

In tegenstelling tot wat nogal eens gezegd wordt kan dit onderwerp uitstekend behandeld worden zonder dat de leerling inzicht heeft in computerhard- en software. In feite hebben ook de meeste gebruikers van computers totaal geen idee van deze zaken. Wie dit verbaast denke slechts aan auto's en TV toestellen, ook ingewikkelde apparaten die door velen gebruikt worden zonder te weten hoe ze werken.

Evenmin is het juist dat de leerling pas interesse voor computertoepassingen kan opbrengen wanneer hij weet hoe het apparaat werkt. Ervaring in de klas toont duidelijk aan dat het leeuwendeel der leerlingen er zeer in geïnteresseerd is te weten hoe je met een computer werkt en er ook inderdaad mee te werken ook al weten ze niet hoe de computer werkt.

Daarom, wanneer men dan al iets aan computer hard- of software wil doen dan dient dit uitsluitend ter bevrediging van de nieuwsgierigheid van de leerling, en staat als zodanig op hetzelfde vlak als het uitleggen hoe een televisietoestel werkt.

Door de voornaamste aandacht aan algoritmiek te geven wordt de omvang van het vak computerkunde tot veel kleinere proporties teruggebracht dan men wel eens geneigd is te denken. En dat is maar goed ook, want op school is tijd een even kostbaar goed als overal elders.

Het is essentieel dat de leerling ook inderdaad met een computer werkt althans in deze zin dat hij programma's schrijft die op een echte computer verwerkt worden. Anders blijft het maar bij droogzwemmen en dit is weinig inspirerend en erg onbevredigend voor de leerling. Pas wanneer hij resultaten van de computer terug krijgt vindt het wonder plaats dat deze machine, die eens met zoveel respect bekeken werd, op zijn juiste plaats gezet wordt: een slaaf van de mens en zelfs van een schooljongen. Een tweede reden waarom het essentieel is dat de leerling programma's schrijft en de resultaten terug krijgt van een computer is om hem gevoel te geven voor de relativiteit van resultaten die met een computer verkregen zijn. Op het ogenblik is het nog zo dat voor velen een bewering, dat de computer iets heeft geconstateerd, geen tegenspraak duldt. In werkelijkheid zijn de resultaten niet van beter gehalte dan degenen die het programma ontwierp. Het is belangrijk dat de leerling zelf ziet hoe gemakkelijk men gevallen over het hoofd ziet en welke grote consequenties kleine programmeerfouten kunnen hebben.

Inmiddels moet men oppassen dat het programmeren niet teveel nadruk krijgt of nog erger dat het hele vak tot een programmeercursus ontgaat. Het is niet nodig dat de leerling een erg goed programmeur wordt. Zijn programma's hoeven geen voorbeelden van efficiency te zijn (althoewel het ook geen voorbeelden van inefficiency hoeven te zijn). Het komt me voor dat een juiste instelling voor de leraar is dat hij het programmeren eigenlijk als een noodzakelijk kwaad beschouwt.

De te gebruiken programmeertaal dient dan ook met grote zorg gekozen te worden. Deze hoeft enerzijds niet al de verfijning van ALGOL of FORTRAN te hebben, maar moet anderzijds ook zeker de leerling niet opschepen met al de administratieve rompslomp van een machinetaal. Er moet prettig met variabelen, rijen en aritmetische uitdrukkingen (bijv. $(a+b)/2$) gewerkt kunnen worden. Om de eerste stappen van de leerling op het computerpad zo eenvoudig mogelijk te maken hebben wij het verantwoord geacht een aparte taal te ontwikkelen, ECOL geheten (Educational Computer Language), die min of meer in de doorsnee van ALGOL en FORTRAN ligt, maar een aantal onaangenaamheden van deze talen mist ten koste van wat efficiency (bijv. is er geen declaratie van enkelvoudige variabelen en er zijn geen types) en die ook wat minder kansen op vergissingen geeft (de slordigheid van leerlingen bij het schrijven van programma's is ongelooft). Het bezwaar dat deze taal ECOL niet algemeen gebruikelijk is en dat de leerling liever een van de bestaande talen zou moeten leren omdat dat hetgene is wat hij later zal nodig hebben, vinden wij niet erg groot: als je eenmaal een programmeertaal kent is het leren van een volgende niet moeilijk meer. Bovendien komen en gaan programmeertalen, en is er alle kans dat de leerling zich later moet gaan bedienen van een programmeertaal die er nu nog niet eens is.

De toepassingen verdienen veel aandacht. Niet alleen de kleine toepassingen die gemakkelijk programmeerbaar zijn, maar ook en vooral de grote toepassingen rondom ons heen waaruit het belang en de interessantheid van computers duidelijk wordt. Dit zijn de toepassingen waarop criteria (a) en (b) van toepassing zijn. Dit ook zijn de problemen waar algoritmiek in zijn ware setting aan het licht treedt, en dit is niet het geval met kleine programmeeropgaven.

Bij deze grotere toepassingen komt men in aanraking met de kwestie van het maken van een *model* van de werkelijkheid. Men kan geen berekeningen maken aan de werkelijkheid zelf, maar de werkelijkheid moet *gerepresenteerd* worden door een computermodel en aan dit model worden de berekeningen gemaakt. Dit heeft twee belangrijke implicaties. De eerste betreft weer de relativiteit van computerresultaten: deze zijn ook niet beter dan het model waaraan de berekening plaats vond. Het model stelt immers hoogstens een *deel* van de werkelijkheid voor en elke invloed die dit deel van de werkelijkheid ondergaat van gebeurtenissen er buiten kunnen niet in de berekende resultaten weerspiegeld worden. De tweede implicatie ligt in het vlak van criterium (d). Een bekwaamheid om relevante modellen van de werkelijkheid te bouwen is belangrijk bij vele onderwerpen op school en daarna. Modelbouw is niet eenvoudig. Het betekent onderscheiden wat belangrijk is (en daarom in het model moet worden opgenomen) en wat niet (en kan worden weggelaten). De aantrekkelijkheid van deze modelbouwwerij in de computerkunde is de nauwe relatie ervan met het alledaagse leven en het feit dat men er geen moeilijke theorieën bij nodig heeft (zults in tegenstelling tot de situatie in bijv. natuurkunde en scheikunde).

Samenvattend: computerkunde bij het AVO moet zich bezighouden met het organiseren van processen met een computer; er zijn dan zeer wenselijke neven-effecten.

4 Schets leerplan voor computerkunde

De voorafgaande beschouwingen hebben ons tot een leerplan geleid dat ik hier slechts beknopt wil schetsen. Nadere uitwerking ervan vindt men in [3].

(i) Elementaire algoritmische begrippen en hun realisatie op een computer.

- (a) geheugen en waardetoekenning
- (b) aritmetische uitdrukkingen
- (c) in- en output opdrachten
- (d) vertakking en samenvloeiing
- (e) cyclus
- (f) rij.

Deze onderwerpen dienen zowel in termen van blokschema's als van een eenvoudige programmeertaal gestalte te krijgen en aan tal van eenvoudige voorbeelden te worden verduidelijkt; door de leerlingen zelf geschreven programma's dienen op een computer verwerkt te worden. Dit deel I zou zoiets als 20 lessen moeten beslaan.

(ii) Algoritmen van hoger plan.

Dit omvat het opstellen van blokschema's voor problemen waarvoor complete programma's te lang zouden worden. De blokken van deze blokschema's stellen elk op zich weer een algoritmisch proces voor waarvan de leerling zich op dit moment kan voorstellen dat er een programma voor te schrijven is of waar-

voor misschien in het voorafgaande al programma's geschreven zijn.

In deze samenhang zou de subroutine ter sprake kunnen komen maar we hebben daarin niet veel nut gezien. De subroutine vergt vrij wat uitleg, en het is moeilijk om voorbeelden te bedenken waarin de subroutine op voor de leerling relevante wijze gebruikt wordt.

Ook is nu het moment aangebroken om wat meer aandacht te geven aan algoritmen die helemaal niet computer-georiënteerd zijn maar aan het dagelijks leven ontleend.

(iii) Principes van belangrijke computertoepassingen.

- (a) personeelsadministratie of burgerlijke stand
- (b) bank- of girostelsel
- (c) reserveringssysteem
- (d) simulatie
- (e) kunstmatige intelligentie
- (f) operaties op teksten (string handling).

Aleen de principes zijn van belang. Een compleet giroprogramma zal wel duizenden opdrachten bevatten. Maar het principe kan uitgelegd worden met een programma van twintig opdrachten. Een meer verfijnd girostelsel kan dan besproken worden met stroomdiagrammen zoals onder (ii) is uiteengezet, programmeeropgaves kan men aan de blokken van deze stroomdiagrammen ontleen en op deze wijze krijgen de leerlingen heel wat inzicht zonder overdedreven programmeerinspanning. Overeenkomstige opmerkingen kunnen gemaakt worden t.a.v. de andere toepassingen. Dit is ook een indicatie dat men de onderdelen (ii) en (iii) tezamen zou kunnen doen.

Dit is klaarblijkelijk een erg bescheiden programma. Toch verwachten we dat dit programma, dat minder dan 2 jaaruur vergt, duidelijk inzicht en bekwaamheid aan een groot aantal leerlingen kan geven. Dit laatst kan zeker niet gezegd worden van vele veel kostbaarder onderwerpen. Dit plaatst computerkunde in een unieke en attractieve positie.

5 Wie onderwijst computerkunde en aan welke leerlingen

Het is een tamelijk controversiële vraag wie de juiste man (of vrouw) is om computerkunde te onderwijzen en ook welke status dit onderwerp in het schoolprogramma moet hebben.

Sommigen menen dat het een apart vak met een aparte docent moet zijn. Naar mijn mening moet dit echter beslist niet. Er moeten niet teveel verschillende onderwerpen op school zijn. Al te lang hebben we gezocht onder een groot aantal vakken en vakjes die duidelijk samen hangen zoals geschiedenis, staatsinrichting en economie; natuurkunde en mechanica; algebra en meetkunde; maar waarvan de kennelijke samenhang niet uit de verf kwam doordat ze als verschillende vakken en door verschillende leraren gegeven werden. Gelukkig is er tegenwoordig eerder een streven naar integratie dan naar differentiatie en dat moet niet door computerkunde doorbroken worden. Juist niet door computerkunde, want als er één vak is dat relaties met vele andere vakken heeft dan is het wel computerkunde.

Dit laatste argument laat velen in de andere richting doorslaan. Men moet het laten geven van computerkundeonderwijs niet beperken tot leraren van een bepaald vak, en in het bijzonder niet tot de wiskundeleraren; wanneer de leraar Frans iets van computers afweet moet hij er maar over vertellen, en idem de leraar scheikunde etc., zo zeggen zij.

Tot op zekere hoogte ben ik het hier wel mee eens. Het is inderdaad zeer belangrijk dat in vele vakken het mogelijke gebruik van computers duidelijk gemaakt wordt, omdat de leerling in zijn latere leven de computer in zo velerlei samenhang zal tegenkomen. Het is ook zeer belangrijk omdat een vak meer operationeel (zie 2) voor de leerling wordt en meer reliëf krijgt wanneer hem de computerimplicaties van het vak getoond worden. Vanzelfsprekend hangt dit in sterke mate af van de toevallige leraar die een vak onderwijst.

De vraag is echter: wie zal de leerling de eerste stappen naar de computer laten zetten? In welk vak zal deze inleiding een plaats krijgen? Want het is beslist ongewenst om deze inleiding te laten afhangen van de toevallige leraren die de verschillende vakken hebben. Hoe eenvoudig deze inleiding ook moge zijn (zie 4), het is een onderwerp met een eigen structuur en methodologie, en iemand die wat van programmeren af weet en weet hoe een computer te gebruiken voor zijn eigen doeleinden is niet ipso facto gekwalificeerd om deze inleiding te onderwijzen.

Het is duidelijk dat van alle vakken op school de wiskunde het vak is dat qua structuur en methodologie het nauwst verwant is met een inleiding tot computerkunde zoals in 4 geschetst. Ook is het waar dat de wiskundeleraar het meest ervaren is om dit soort manipulaties en structuren te onderwijzen en daarmee ook om gebrek aan begrip bij de leerlingen op te sporen en te verhelpen.

Het onderwijs in de principes van de toepassingen zie ik graag in een hand, om de daarbij optredende samenhang aan het licht te brengen, en weer is het de wiskundeleraar die ik dit het liefste zie doen. Hij houdt op waar het interessant

begint te worden voor de professional in wiens vakgebied de toepassing ligt. Dit opent de mogelijkheid voor een gelukkige samenwerking en taakverdeling. Het zou echter ook heel goed zijn voor het wiskundeprogramma zelf als de computerkunde daarin een plaats kreeg. Het wiskundeprogramma immers neigt steeds meer in een abstracte richting; deze realiteitsinjectie kan een sterke impuls in een meer toegepaste richting geven.

Om deze redenen ben ik van mening dat het gerechtvaardigd is te trachten de computerkunde in het wiskundeprogramma onder te brengen.

Hierbij rijst natuurlijk vanzelf de vraag of dan de wiskunde extra uren moet krijgen, en waar die vandaan moeten komen. Ik vraag me evenwel af of dit inderdaad nodig is.

Eenzijds dient men hierbij te overwegen dat de wiskunde altijd voor een belangrijk deel gemotiveerd is door zijn vormende waarde. Computerkunde heeft een vormende waarde die deels in hetzelfde vlak ligt als die van de wiskunde, maar de vormende waarde van computerkunde komt naar mijn overtuiging beter tot zijn recht door de veel grotere mogelijkheden voor creatief bezig zijn die dit vak de leerlingen biedt.

Voorts werd de wiskunde vooral gemotiveerd door haar toepasbaarheid. In diverse toepassingsgebieden heeft de computer evenwel de wiskunde verdrongen. Een en ander maakt dat een claim van de computerkunde op wiskunde uren zeker overwogen dient te worden, mede in het licht van de opmerkingen over optimalisatie aan het eind van 1.

Anderzijds zou het mij niet verwonderen wanneer computerkundeonderwijs een zodanig gunstige invloed op de instelling van de leerlingen heeft (zie hierboven onder criterium (d)) dat, ook al neemt men de tijd voor computerkundeonderwijs uit de wiskunde-uren, toch het wiskundeprogramma niet hoeft te worden ingekrompen. Er zijn reeds enkele positieve indicaties in deze richting. Het zou zeer wenselijk zijn wanneer de leraren die nu reeds met computerkunde experimenteren, over dit punt hun licht eens zouden willen laten schijnen.

Resumerend: het is zeer gewenst dat de wiskundeleraar de inleiding geeft, en dat alle andere leraren maximale aandacht aan het verschijnsel computer schenken in hun lessen.

Tenslotte nog de vraag aan wie computerkundeonderwijs gegeven moet worden. In het voorafgaande ligt besloten dat alle leerlingen dit onderwijs moeten krijgen en wel in een niet al te laat leerjaar, dit laatste vooral in verband met de vormende waarde van het vak. Wij denken aan hoogstens de derde klas van het AVO.

6 Ervaringen

De ervaringen in de klas zijn nog gering. Wat we tot dusverre hebben gezien stemt ons echter optimistisch. De leerlingen zijn in grote meerderheid enthousiast, en komen tot niet vermoede activiteiten. Eigen problemen worden bedacht, en er ontstaat teamwork. Wij zouden het zeer op prijs stellen als ervaringen van tijd tot tijd een plaats in dit tijdschrift zouden vinden.

LITERATUUR

- [1] Rapport over de wenselijkheid en mogelijkheid van het invoeren van computerwiskunde in het onderwijs voor MAVO, HAVO en VWO. Commissie Modernisering Leerplan Wiskunde, Utrecht 1968.
- [2] Computers in higher education. Report of the President's Science Advisory Committee, Washington 1967 (het zgn. Pierce report).
- [3] C. A. C. Górts, S. G. van der Meulen, A. van der Sluis, J. R. Zweerus: Computerkunde voor AVO en VWO. Wolters-Noordhoff, Groningen 1970.
- [4] A. van der Sluis: Onderwijs in computerkunde. In: J. H. Wansink, Didactische Oriëntatie voor wiskundeleraars, deel III, p. 276-315, Wolters-Noordhoff, Groningen 1970.

1.2 VOORTGEZETTE COMPUTERKUNDE

1.2.1 B a s i s c u r s u s

In de voorgaande paragraaf is het hoe en waarom van het leervak computerkunde uiteengezet. Dit betreft een algemeen vormend leervak dat, volgens 73% van de deelnemende docenten in 1971, twee werkelijke lesuren gedurende één leerjaar zou moeten beslaan. Kort samengevat zijn de belangrijkste oogmerken van deze cursus:

- voorbereiding op de samenleving, waarin de computer een steeds toenemende rol zal spelen. Het is aan ons een generatie op te voeden, die zal zorgen dat dit een dienende rol zal blijven.
- voorbereiding op toekomstig werk, met het besef welk deel van de werkzaamheden aan computers kan worden overgelaten.
- voorbereiding op toekomstige studie, waarin vrijwel zeker computertoepassingen op wetenschappelijk terrein voorkomen.
- aankweken van een operationele instelling, d.w.z. een instelling waarmee kennis ook geschikt gemaakt wordt om mee te werken.
- aankweken van algoritmische vaardigheid, d.w.z. het vermogen om schijnbaar chaotische systemen te ordenen rond routinematige kernen.
- aankweken van organisatorische kwaliteiten, d.w.z. het onderscheiden van niveaus van complexiteit, het voorzien van alle mogelijke gevallen en het onder één noemer brengen van schijnbaar verschillende zaken.
- Inzicht verstrekken in het bouwen van een model, geen berekeningen makend aan de werkelijkheid zelf, maar aan een computermodel dat deze werkelijkheid representeert.

In de doelstellingen die tot nu toe zijn uiteengezet wordt vermeld dat het leervak bedoeld is voor *alle* leerlingen van het algemeen voortgezet onderwijs. Ik zou daar graag het lager beroeps-onderwijs aan willen toevoegen. Of dit leervak apart of geïntegreerd in wiskunde, praktische be-roepsvorming of welk ander vak ook, moet worden gegeven, wordt verschillend beoordeeld. Opgenomen in bijvoorbeeld de wiskunde heeft het bezwaar dat computerkunde niet de haar toegemeten tijd krijgt. Te veel bij een leraar met computerhobby, te weinig bij een overladen programma. Een apart leervak computerkunde druist in tegen de moderne opvattingen van integratie van leer-vakken. Vraag is in dat geval ook welke leraar bevoegd moet worden geacht voor het doceren van computerkunde.

1.2.2 V o o r t z e t t i n g v a n d e b a s i s c u r s u s ?

Bij veel docenten komt de vraag op of na beëindiging van de basiscursus het leervak niet kan worden voortgezet in de hogere leerjaren, mogelijk zelfs een eindexamenvak kan worden. Dit wordt mede gevoed door het feit dat in veel gevallen binnen het experiment de computerkunde gedoceerd werd als keuzevak in de vierde klas HAVO. Andere keuzevakken kunnen vaak voortgezet worden in de vijfde klas HAVO.

Welke inhoud zou moeten worden gegeven aan deze voortzetting? Bij de beantwoording van deze vraag zijn m.i. zes aspecten te onderscheiden. Van de eerste vijf kunnen combinaties voorkomen, maar hieronder worden ze zoveel mogelijk disjunct aangegeven. Getracht wordt aan te tonen dat alleen het zesde aspect aanvaardbaar is voor algemeen voortgezet en lager beroepsonderwijs.

1) **Apparatuur** (hardware)

De constructie en de elektronische werking van de computer, bijvoorbeeld de werking van een kernengeheugen, binaire code, poorten, logische schakelingen.

Deze leerstof heeft voor vele docenten met een beetje technisch gevoel, en ik reken mezelf hier bij, een grote charme. Wiskundeleraars die weleens hun natuurkunde-collega's hebben benijd om de mogelijkheid hun leerlingen in een praktikum te laten werken, zien nu een mo-gelijkheid om met 'computerbouwdozen' hiermee te wedijveren.

Vragen we ons af welk doel we dienen met deze leerstof, dan moeten we meestal het antwoord schuldig blijven. De kans dat we een toekomstig computertechnicus onder onze leerlingen tellen is minimaal. Kennis die de leerlingen hebben opgedaan is niet toepasbaar en is in alle gevallen een zo sterke simplificatie, dat de werkelijkheid nauwelijks meer gedekt wordt. Als U vindt dat Uw leerlingen toch wat Boole-algebra hebben opgestoken, dan hebt U het paard wel achter de wagen gespannen.

Willen Uw leerlingen (althans technisch aangelegde jongens) zo graag weten hoe het apparaat werkt?

Maar ook de meest kritische leraar zal zijn lessen niet alleen afstemmen op wat zijn leerlingen het liefst horen. En, nogmaals, de constructie van een computer kan niet worden behandeld zonder de leerlingen een rad voor de ogen te draaien.

2) *Harde programmatuur*

Dit is het programmapakket dat altijd in de computer aanwezig is om het apparaat prettig bruikbaar te maken. In werkelijkheid zal de opdracht `a := LEES` een programma activeren, dat een getal cijfer voor cijfer leest, een toestandsteken en eventuele decimaalpunt op de juiste wijze interpreteert en het getal opbergt in de bedoelde geheugenplaats. Geen enkele gebruiker van een computer maakt zich het bestaan van harde programmatuur voortdurend bewust. De computerleverancier dient zich hierover zorgen te maken. Het is niet de bedoeling van ons onderwijs om personeel van computerfabrikanten op te leiden. Deze leerstof zal vrijwel niet kunnen bogen op belangstelling van de leerlingen.

3) *Feitenkennis over computers.*

Hiermee bedoel ik feiten als: een bepaald type computer heeft een kerngeheugen van 32 K woorden van 27 bits, men onderscheidt cilinderdrukkers, kettingdrukkers, letterstaafdrukkers, de draaisnelheid van een schijfengeheugen enz.

Deze leerstof is natuurlijk gegarandeerd verouderd wanneer Uw leerling de school verlaat. Deze mogelijkheid noem ik hier speciaal omdat, tot schande van het onderwijs binnen het koninkrijk, de eerste poging tot het afnemen van een eindexamen in computerkunde juist dit soort leerstof betrof.

4) *Voortgezet programmeren.*

De behandeling van subroutines, recursiviteit, programmeren met een stapel, datastructuren, enz.

Hiermee is geen gemakkelijke maar zinvolle vulling gevonden, met vele mogelijke toepassingen. Handel en industrie tonen belangstelling voor een dergelijke opleiding, om de duidelijke reden dat men op de programmeursopleiding hoopt te besparen. Deze cursus voorziet in een gedeeltelijke beroepsopleiding voor programmeur en dat is alleen voor middelbaar en hoger beroepsonderwijs te verdedigen.

5) *Toepassingen uit andere leervakken*

Het is heel goed denkbaar dat de docent computerkunde toepassingen behandelt van computergebruik in andere leervakken, zoals wiskunde (inklusief waarschijnlijkheidsrekening en statistiek), natuurkunde, biologie, economie, boekhouden, aardrijkskunde, literatuuronderzoek.

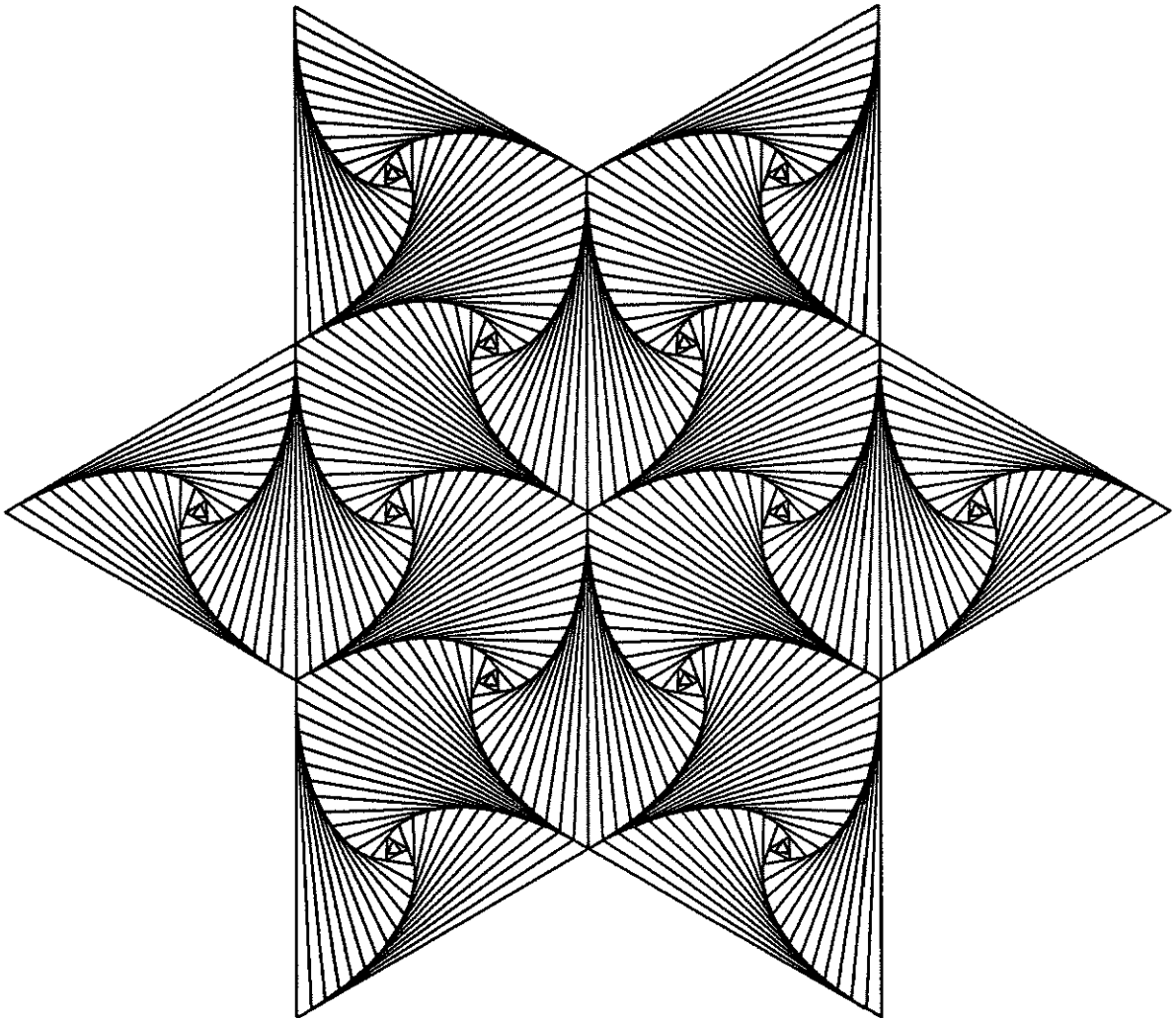
Hierbij rijzen twee bezwaren. Welke leraar computerkunde, hoe breed zijn achtergrond ook moge wezen, zal zich thuis voelen in al deze disciplines, om hierover gefundeerd te kunnen spreken? Verder zal Uw gehoor bestaan uit een groep leerlingen met totaal verschillende keuzepakketten. Snijdt U een natuurkundig onderwerp aan, dat zullen de leerlingen die dit vak niet gekozen hebben, weinig begrip en belangstelling tonen. Zo zal elk onderwerp dat U kiest maar een gedeelte van Uw groep kunnen boeien. En ik weet uit eigen ervaring, dat dit zacht is uitgedrukt.

6) *Computerkunde toegepast in andere leervakken*

Het voorgaande punt bevat wel de sleutel tot de oplossing die ik zou willen aanbevelen. Als een computertoepassing in bijvoorbeeld de natuurkunde behandeld wordt in de natuurkundeles, dan is het natuurkundige aspect zeker duidelijk aan de toehoorders of kan duidelijk gemaakt worden door de terzake kundige docent. U werpt nu twee bezwaren op.

Ten eerste: zijn alle leerlingen in de natuurkundeles op de hoogte van programmeren? Als antwoord hierop moet misschien nog 'nee' gelden, maar 'ja' is toch wel het ideaal, zoals U kunt lezen aan het eind van punt 5 van § 1.1. Ten tweede: is de leraar natuurkunde (of welk vak ook) op de hoogte van de mogelijkheden die de computer biedt voor zijn vak? Veel recentelijk afgestudeerde collega's zijn dat wel, maar in de meeste gevallen zullen we ook hierop ontken-
nend moeten antwoorden.

Als U na lezing van het voorgaande overtuigd bent, dat facet 6 de ideale situatie aangeeft, probeert U Uw collega's dan te wijzen (en warm te krijgen voor) de mogelijkheden, die de computer ook in hun vak biedt.



1.3 INTERNATIONALE AANBEVELING

Van de 'Organisation for Economic Co-operation and Development' (OECD) is in 1971 een aanbeveling uitgegaan naar alle regeringen, met richtlijnen voor een inleidende cursus computerkunde. Deze richtlijnen zijn opgesteld door een werkgroep, waarin van de 42 lidlanden er 8 (waaronder Nederland) vertegenwoordigd waren. Alleen een gedeelte van wat handelt over de minimale inhoud van de cursus is hieronder overgenomen. Aan Module 2, II (a)(3) is voor de duidelijkheid toegevoegd het hier ten lande meer bekende SERA.

Module 1: Automata, Algorithms, Flowcharts

Introduction to automata: i.e. devices that are able to perform a few well-defined operations. The order in which the operations are performed is vital. Examples: juke-box, washing machine, control unit of an elevator, knitting (human automaton). Also of importance is the fact that instructions to an automaton, as distinct from human beings, should be unambiguous and complete. The fundamental idea of an algorithm to be used in this course is simply an ordered set of instructions. Algorithms can be described by flowcharts and consequently some skill must be developed in flowcharting. It is a waste of time to give too much attention to linear flowcharts. From the early stages branching and looping should be used.

Examples of situations which give rise to interesting algorithms, and are suited to be followed-up with simple exercises are:

- a) Human activities as putting stamps on letters according to value of stamp and destination of letter, two people playing a hazard game for money, changing wheels of a car from front to rear with a jack and a spare wheel, etc.
- b) Administration (payroll)
- c) Booking policies (e.g. airplanes)
- d) Inventory policies (e.g. supermarket ordering goods)
- e) Sorting
- f) Selection of library cards
- g) Gregorian calendar
- h) Arithmetical operations that are constructed from simpler ones.

The chosen examples should be mostly of a non-mathematical nature. The instructions which are used should be on a verbal level of the type: 'take next card, put on a 25c stamp, is the light off?' etc... They will generally differ

from problem to problem, and it is advisable to provide the pupils with the standard instructions to use. (if they must find them themselves pupils generally tend to expand into irrelevancies).

It should be pointed out that carrying out a given algorithm is just routine work, very much suited to an accurate but stupid automaton, whereas devising algorithms is by no means routine - it is a highly creative job.

It could be tempting in this module to treat sophisticated algorithms (each of the examples a-f could give rise to that) but this should be resisted. Such examples can be taken up in the second set of modules on applications, after some programming skill has been acquired and they will thus function in a proper setting.

On the other hand some of these examples could be used to illustrate the operations of a computer. This would require that the examples be translated into numerical terms and that the pupils would need to use the operations of addition, subtraction, comparing, etc. and in fact move closer to the instructions used in a computer.

Module 2: Programming

I. Introduction

In this module the pupil will make contact with the computer and will learn the language which allows communication with it. In consequence it is important that, particularly in the early stages the pupil is given a general understanding of the structure of the computer and its different parts (central processor, memory, input, output devices, control unit). This should arise naturally when programming is introduced and indeed will be essential for the proper understanding of the various operations. Programming is only one step in the solution of a problem. The complete sequence of events is:

- 1) Analysis of problem
- 2) Preparation of flowcharts
- 3) Preparation of programme
- 4) Data preparation
- 5) Testing and running of programme
- 6) Results

The most important stages in the solution of a problem are the first two which:

- a) define the problem
- b) define the solution.

Programming merely expresses the solution in a form suitable for computer interpretation. Pupils should therefore be encouraged to draw flowcharts which define the solution to the problem and using these as precise plans hence produce the set of programme instructions. This approach ensures that the pupil when writing a programme is concentrating solely on the problems of language and is not side-tracked by the logic of the solution. Some experience of programming is essential although it is not necessary to spend a great deal of time on it. Neither is it necessary for all pupils to write more than a very few simple programmes.

II. Programming Languages

There are a large number of programming languages available but it should be noted that not all languages are available on all computers. Consequently, the choice of language must depend largely on what facilities are available since no language however desirable should be used unless the programmes written by the pupils can be run. It should be noted that whichever existing language is used there will be undesirable teaching features.

There are two main types of programming languages:

a) Machine Dependent Languages

These are programming languages the nature of which is determined by the structure of a particular computer.

- 1 - Machine Code
- 2 - Assembly Language
- 3 - Language representing an imaginary machine (e.g. MIX, SERA)

Because of the unwieldiness and the technical and machine dependent nature of these languages, it is not recommended that they be used in this course for programming applications.

b) High Level Languages

Such languages are designed to cope with a specific range of applications rather than for a particular computer. A programme written in a high level language can (usually, with minor modifications) be run on different computers.

If a teacher is offered facilities using a high level language, it is hoped that he will use the language very simply and not take advantage too early in the course of all the high level features.

III. Development

Irrespective of which high level programming languages, or subset of a high level programming language is used the development in each case should be gradual with the introduction of one language feature at a time.

Stage (i): Simple programmes to add, subtract, multiply and divide introducing ideas of assignment, input, output, comparing, branching, counting.

Use should be made of some of the examples of Module 1 but they must be kept simple.

At this point, a minimum amount of programming has been covered so Module 3 could follow but teachers who choose to end their work on programming here will find that they are limited to a small range of fairly trivial examples when they work Module 3.

Stage (ii): (if desirable).

More branching, cycling, one dimensional arrays, files.

Module 3: Computers and Society

This section of the work provides the *raison d'être* for the whole of the introductory course. As has already been pointed out, many aspects of the use of computers in society will have been used to motivate the earlier work on programming, but some facility in communicating with the computer having been established, it should now be possible to

discuss with greater understanding the various implications of computer use. These implications - political, social, ethical, economic and legal - must where appropriate be discussed together with questions such as 'who takes decisions?'. In this context pupils must be led to appreciate that the computer on its own does not decide but rather that the decisions are still human responsibilities and that these are taken at various levels by programmers, systems analysts, administrators, etc.

It is impossible to provide an exhaustive list of computer uses. Immediately obvious are such applications as payrolling, accounting, billing and so on, and indeed pupils may have already treated these at an elementary level in their work on programming. From these types of situations, the pupils should be led to various aspects of computer control. These can come from situations such as: traffic control, that is traffic lights under computer control; credit control, that is the movement towards a society where credit cards will replace money; pollution control, now practised in many industrialised countries; production control, such as steel rolling; and finally the control which could be exercised by the use (or misuse) of data banks.

Pupils should be required to make a case study at some depth on one such topic. Topics which are not covered at depth can be used for discussion with pupils or can be the motivation for visits to various computer users. In addition to the applications already suggested, it could be fruitful for pupils to look at possible future applications. This could include the use of the computer in the fine arts, and could also include some introduction to game playing on the computer.

The work in this module would benefit greatly from the use of a variety of supporting materials. These would include books, wall charts, slides, film loops, and films. Some indication of appropriate material will be given in a document to be published separately.

2 Aanwijzingen voor de les

2.1 DE LEERSTOF

2.1.1 Het blokschema (Compk § 3.1, Werks hfdst. 1)

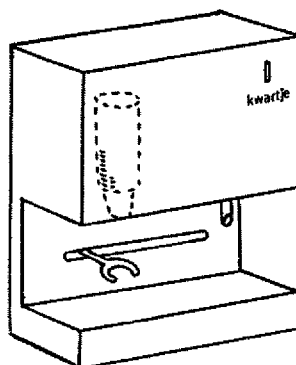
Waarom moeten blokschema's behandeld worden? Men hoort vaak protesten dat men de leerling de programmeertaal kan bijbrengen zonder over blokschema's te spreken. Dat is juist. Maar men bedenke dat in onze doelstelling niet uitsluitend het leren programmeren staat vermeld. Belangrijker is het leren analyseren van een probleemstelling. En daarvoor is het blokschema een aangewezen, hoewel mogelijk niet het enige, expressiemiddel. Voor U als docent is verder van belang dat U in Uw uitleg altijd terug kunt vallen op het uitbeelden in blokschema's. Een leerling die bij het maken van een wat ingewikkelder programma steeds fouten maakt kan, zonder hulp van blokschema's, haast niet op het juiste spoor gezet worden.

Het blokschema wordt gepresenteerd als middel om een proces te beschrijven. Met opzet zijn in de voorbeelden en opgaven processen gekozen uit het dagelijks leven. De overgang voor de leerlingen naar dit nieuwe leervak gaat daardoor zonder drempel en het werkt animerend om in de wereld om ons heen processen te herkennen. Een eenduidig voorschrift om een proces uit te voeren met eindig veel handelingen uit een repertoire van welbegrepen en uitvoerbare handelingen, wordt een *algoritme* genoemd.

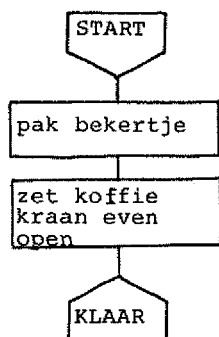
Geleidelijk kan men overgaan op algoritmen voor een automaat. Als voorbeeld hiervan het volgende klasseggesprek (naar aanleiding van Compk opgave 3.1).

docent

We stellen ons een koffie-automaat voor. Er is een gleuf voor kwartjes. In het inwendige is een stapel bekertjes. Een klauwtje kan een bekertje onder een tuit houden. Uit deze tuit kan bruin vocht stromen, dat goedwillende mensen ook wel koffie noemen. Schrijf een blokschema dat de werking van de automaat aangeeft nadat er een kwartje is ingeworpen.



leerling



Dit betekent: er valt één beker van een stapel in een klauwtje. Het klauwtje schuift onder de tap. Heet water stroomt net lang genoeg door oploskoffie om een beker te vullen

maar aan dat er altijd electriciteit en water is. Wat kan er fout gaan als de automaat stipt dit schema opvolgt.

klas

Het kan voorkomen dat er geen bekertjes meer zijn.

docent

Verbeter het blokschema met de vraag 'Zijn er nog bekertjes?'.

leerling

docent Wat kan er fout gaan ?

leerling De stroom kan uitvallen.

docent

(Realiseert dat zijn vraag te vaag was). We gaan een blokschema maken voor wat er in de automaat gebeuren moet. Moeilijkheden buiten de kast kunnen we niet verhelpen. We nemen nu

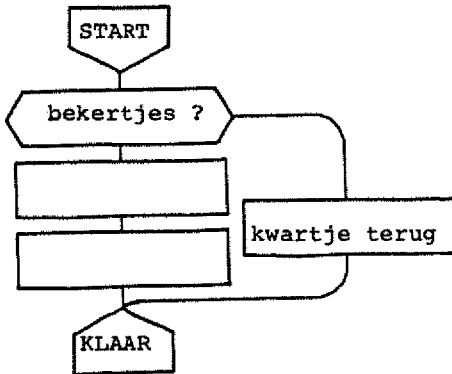


docent

Heeft iemand bezwaar tegen een automaat die werkt volgens dit schema ?

leerling

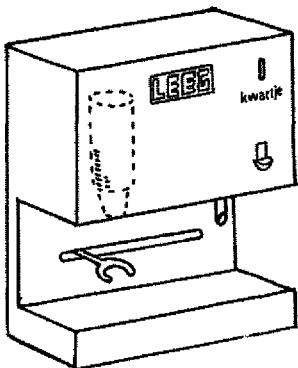
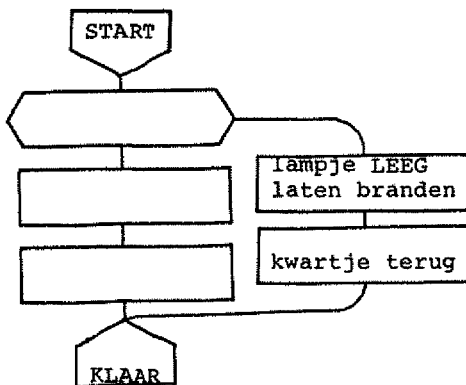
Als de bekertjes op zijn, ben je je geld kwijt zonder iets te krijgen. Het schema moet zijn:



docent

Deze automaat is wel onduidelijk. Wanneer er geen bekertjes meer zijn, komt het kwartje onmiddellijk terug. De klant zal het dan nog eens met een ander kwartje proberen. Steeds zonder resultaat. We zullen daarom de automaat uitbreiden met een lampje met 'LEEG' er op. Geef dit in het blokschema aan.

leerling



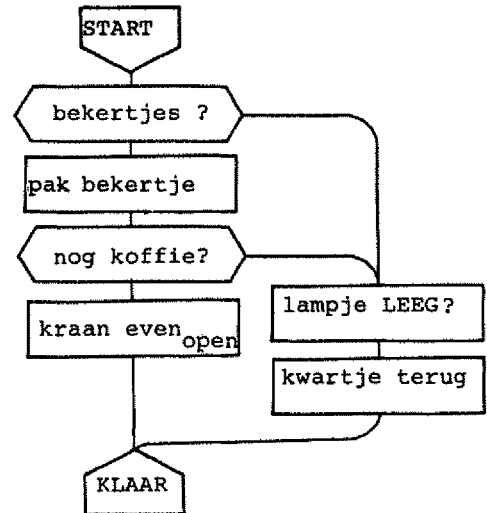
docent

Is er nog een probleem waarin ons blokschema niet voorziet?

klas De koffie kan ook nog op zijn.

docent

Ik zal het schema veranderen.



Is het zo goed ?

klas (5 minuten verbaasde stilte)

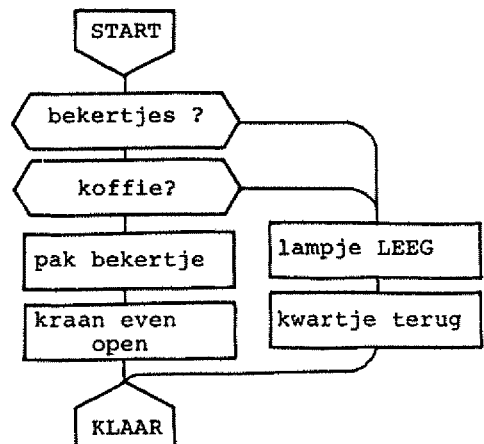
leerling

Als er wel bekertjes zijn en geen koffie en iemand gooit steeds opnieuw zijn kwartje er in, dan valt er steeds een nieuw bekertje in het klauwtje. Op den duur zal de boel klem lopen door de stapel bekertjes.

docent

Goed zo. We moeten er nu eenmaal rekening mee houden, dat de automaat geen verstand heeft. Dus moeten wij ons verstand wel gebruiken om het blokschema ook in deze gekke situatie te laten voorzien. Verbeter het blokschema.

leerling



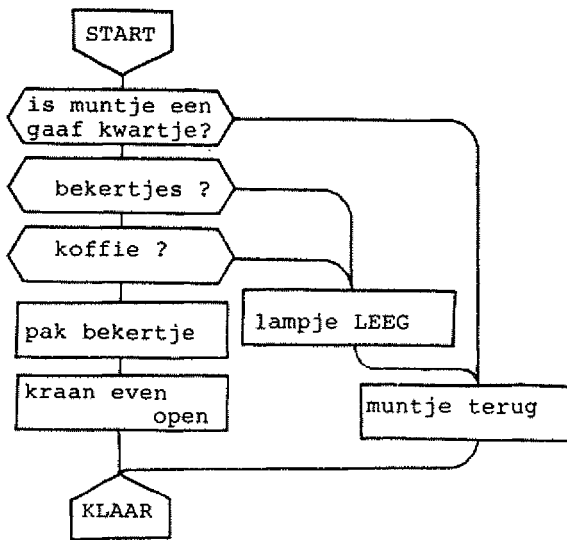
docent

Zijn er nog meer voorzieningen die we willen treffen ?

klas Moet het kwartje nog gecontroleerd worden?

docent Probeer maar.

klas (na enig puzzelen)



kommentaar

In een klasseggesprek is het mogelijk om tijdens de oplossing af te spreken, welke vragen of opdrachten nog wél en welke niet 'verstaanbaar' zijn voor de automaat. Een opgave waarbij de leerling zelfstandig een blokschema moet opstellen, mag nooit zo vaag gesteld zijn als bijvoorbeeld: 'maak een blokschema voor een koffieautomaat'. Een leerling die als oplossing geeft:



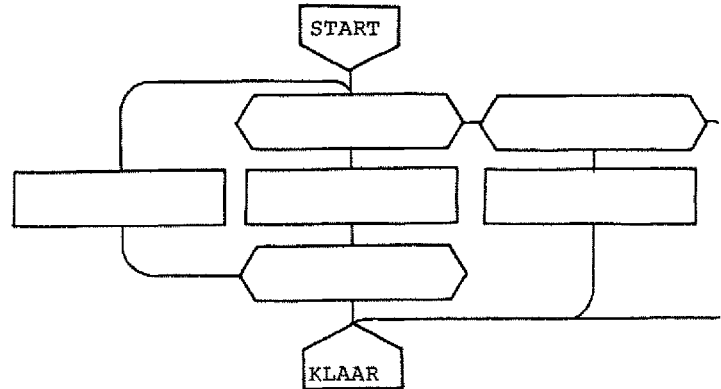
heeft dan in wezen gelijk. De kans is echter nog groter dat de leerlingen zich uitputten in de kleinste detailhandelingen. De leerlingen moeten bij de opgave dus altijd voorzien worden van een verzameling van elementaire vragen en opdrachten die voor de automaat (in ruime zin) nog net verstaanbaar zijn (zogenaamde 'primitiva'). Vergeet U bij Uw blokschemavraagstukken ook nooit de verzameling primitiva te geven.

De besproken opgave luidde met opzet: de werking *nadat* een kwartje (of muntje) is ingeworpen. U kunt zich beter niet inlaten met blokschema's als:

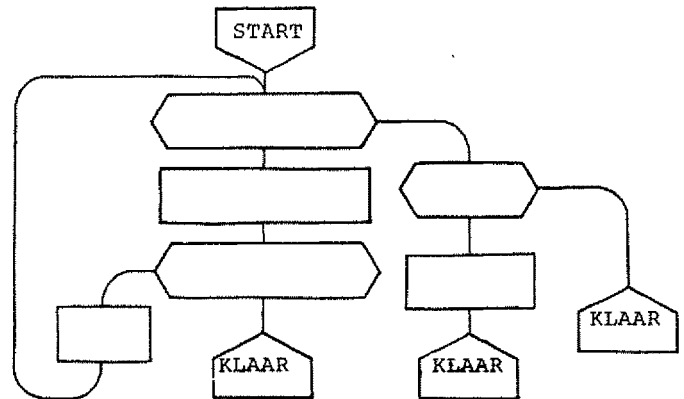


Dit is een cyclus waarbij we moeten veronderstellen, dat het beantwoorden van de vraag tijd kost. Maar hoeveel tijd?

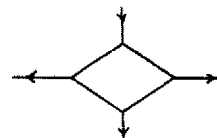
Men lette speciaal op het naleven van conventies bij deze blokschema's. De 'ingang' van elk blok is midden boven. Let op beide, veel voorkomende, fouten in onderstaand schema.



Ook zal men slechts één KLAAR-blok per schema toestaan. Dus niet dit:



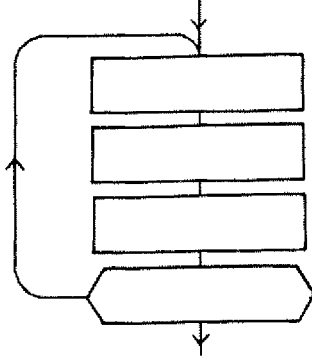
Deze conventie wijkt af van de inmiddels in Nederland genormaliseerde vorm. De laatste beveelt bijvoorbeeld voor het vraagblok aan:



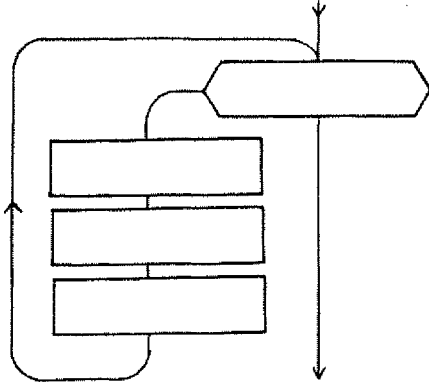
waarbij drie uitgangen tegelijk gebruikt mogen worden, zonder dat de betekenis vastligt. De ruit heeft het bezwaar gauw groot genomen te moeten worden bij veel tekst. Het twee uitgangen-principe in onze leerboeken werd verkozen in verband met de programmeertaal. Dat de ja-uitgang altijd op dezelfde plaats (midden onder) zit is geen enggeestigheid, maar werkt op den duur alleen maar prettiger. Zou dat niet zijn afgesproken, dan moet bij elke uitgang van een vraagblok 'ja' of 'nee' worden bijgeschreven. Iets wat leerlingen al gauw verwaarlozen.

2.1.2 Blokschema's met herhalingen (Compk § 3.2, Wrks hfdst. 6)

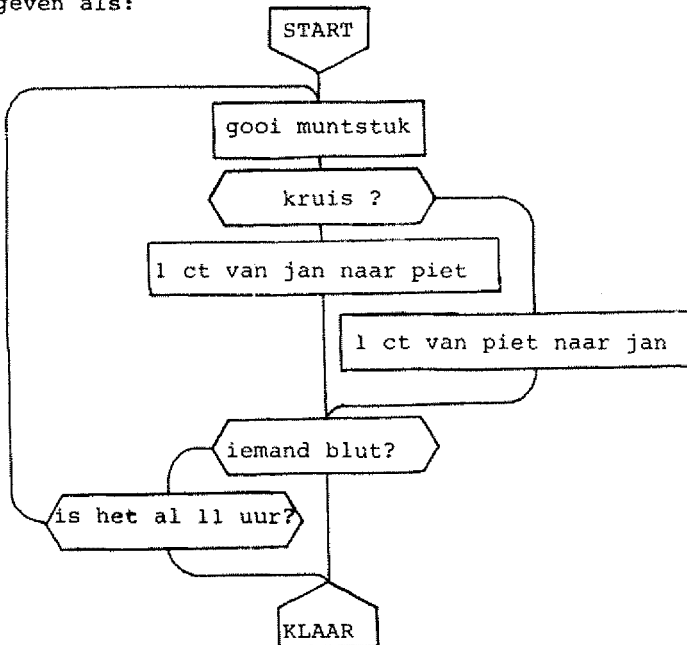
In de tekst wordt gewezen op de noodzakelijkheid van een vraagblok (splitsing) in een cyclus. Uw leerlingen zullen vanaf dit moment voortdurend gewezen moeten worden op het gevaar van een eindeloze cyclus. Blokschematisch zal een goede cyclus iets van de volgende gedaante hebben.



In veel gevallen is het raadzaam om het vraagblok 'eerder' in het schema op te nemen, bijvoorbeeld:

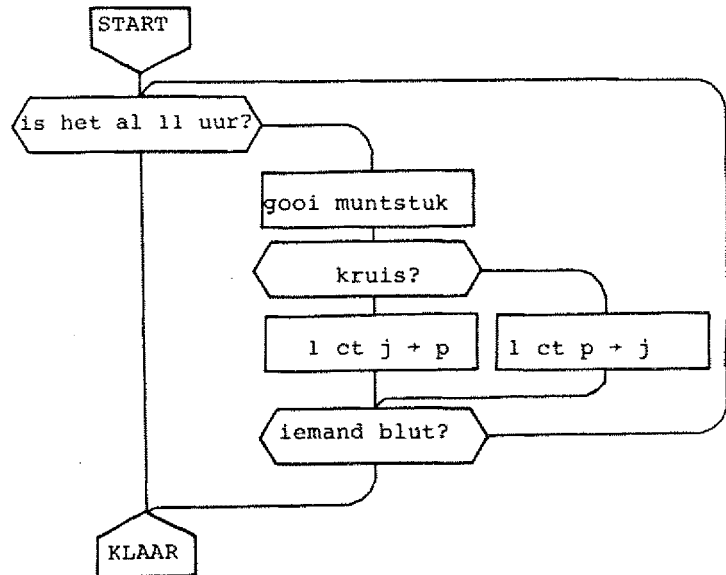


De oplossing van Compk opgave 3.9 wordt vaak gegeven als:



Als U echter bedenkt dat Jan en Piet wel eens na 11 uur het idee kunnen krijgen om te gaan gokken,

dan zou U dat waarschijnlijk willen verbieden. Volgens bovenstaand blokschema wordt er toch één maal kruis-of-munt gegooid. Dit wordt voorkomen in onderstaand schema.



Ook de blutvraag zou boven het muntstuk gooien geplaatst kunnen worden, maar dat is minder urgent zolang bekend is dat beide gokkers met 10 ct beginnen.

2.1.3 Programmeren

Na het eerste hoofdstuk wordt een begin gemaakt met het programmeren. Een programma is een algoritme, geschreven in een vorm die, na eventuele tussenbewerkingen, geschikt is voor een computer. Met de tussenbewerkingen wordt bedoeld: het verpensen op kaarten of papierband of het aanschrappen op kaarten.

Waarom programma's schrijven? Is het niet voldoende dat door een leerling is geanalyseerd en dat hij de oplossing in blokschema heeft aangegeven? Daartegen kunnen drie argumenten worden ingebracht.

- 1) De leerling moet kennismaken met de hele weg die een probleem aflegt om op een computer verwerkt te worden, zodat rondom de computer zo weinig mogelijk geheimzinnigheid blijft bestaan.
- 2) Het is voor leerlingen bijzonder stimulerend om ook resultaten te krijgen uit de computer, na al het voorbereidende werk wat hij er aan heeft verricht.
- 3) Door de computer wordt er de nodige controle op het werk verricht. Hij krijgt de ervaring dat een vrijwel foutloos algoritme bijzonder storende fouten in de resultaten kan veroorzaken. Argumenten als 'het komt uit de computer, dus is het zo' zijn hem hierna vreemd.

2.1.4 O_n_v_e_r_t_a_k_t_e___r_e_k_e_n_s_c_h_e_m_a'_s (Compk hfdst. 5, Wrks hfdst. 2)

De blokschema's in het bedoelde hoofdstuk zijn zo eenvoudig, dat ze niet eenvoudiger of prettiger leesbaar zijn dan de overeenkomstige programma's. De illustrerende waarde van blokschema's gaat pas waarde krijgen, wanneer vertakkingen optreden. Toch is hier het blokschema gehandhaafd uit continuïteitsoverwegingen.

Dit hoofdstuk is een eerste kennismaking met een vast handelingenrepertoire, in tegenstelling tot het voorgaande hoofdstuk, waarin in elke opgave de toegestane verzameling (vragen en) opdrachten anders was.

Geheugen

Men wijze de leerlingen erop dat beslist niet het menselijk begrip geheugen bedoeld wordt. Het computergeheugen is uitsluitend geschikt voor bewaren van getallen (ook al kunnen deze als opdrachten geïnterpreteerd worden). In het menselijk geheugen wordt informatie teruggevonden door associatie, in het computergeheugen door plaatsbepaling. Deze geheugenplaatsnummering of adressering wordt in de programmeertaal vervangen door het gebruik van geheugenplaatsnamen.

In andere programmeertalen is er soms een andere schrijfwijze voor het wordt-teken, zoals "=" (kan verwarring wekken met het algebraïsch gelijkteken, zoals in $a = a + 2$) of "+".

Het laatste teken geeft wel duidelijk aan dat het rechter lid eerst berekend wordt en daarna het resultaat geplaatst in de geheugenplaats die links wordt aangegeven.

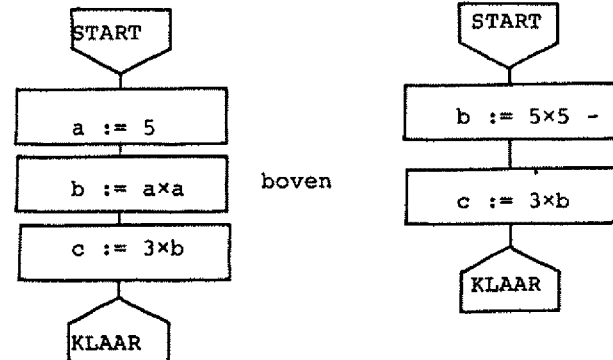
Punten die verder de nadruk verdienen zijn:

- het lezen van een getal uit een geheugenplaats houdt niet in dat het getal daarna van die plaats verdwenen is.
 $b := a$ is een opdracht om te kopieëren, niet om te verplaatsen. De zinsnede 'haal een getal uit een geheugenplaats' is misleidend.
- een geheugenplaats is nooit leeg. Men mag er niet op rekenen dat een in het programma nog niet gevulde geheugenplaats automatisch een nul zou bevatten.
- het werken met geheugenschema's (of kaartenbakgeheugens) moet worden gestimuleerd.

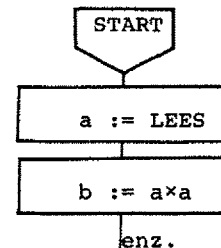
Als het tijdens Compk § 5.3 of Wrks hfdst. 2 te

pas komt, kan de schrijfofdracht al wel eerder worden ingevoerd.

Het is mogelijk raadzaam een (niet tegenover de leerlingen uitgesproken) voorkeur te hebben voor:



voor het berekenen van $3a^2$. Het linker blokschema geeft namelijk later een betere overgang naar:



2.1.5 D_e_z_e_l_f_d_e___g_e_h_e_u_g_e_n_p_l_a_a_t_a_a_n___b_e_i_d_e___z_i_j_d_e_n___v_a_n___h_e_t_w_o_r_d_t_-t_e_k_e_n (Compk 5.4, Wrks vanaf 7.6)

Voor een juist begrip voor dit soort opdrachten is het oefenen met geheugenschema's voor de leerlingen onontbeerlijk.

In Compk 5.4 wordt gesproken over het bezuinigen op het aantal geheugenplaatsen. Het is niet raadzaam om het bezuinigen op het aantal geheugenplaatsen in willekeurige programma's aan te moedigen. De kans op programmafouten neemt alleen maar toe en de leesbaarheid van het programma af.

Naar aanleiding van Compk opgave 5.16 het volgende.

De werking van de blokschema's moeten worden gecontroleerd. Laat U dat de leerlingen doen met twee getalwaarden voor a en b. Bijvoorbeeld 20 en 3. Voor Uw gevoel zal dit misschien weinig bewijskrachtig zijn. Maar het is beslist te ontraden om een bewijs te leveren door in

a := a+b
b := a-b
a := a-b

de begingetallen voor te stellen door bijvoorbeeld p en q.

a	b
p	q
p+q	q
p+q	(p+q) - q = p
(p+q) - p = q	p

De leerling zal de rest van de cursus het idee houden dat er op deze wijze letters kunnen worden opgeborgen in geheugenplaatsen.

Als oplossing van Compk opgaven 5.17 en 5.18 kunt U drie verschillende beelden verwachten:

- de zelf gekozen getallen komen door de hele berekening voor. Een op zichzelf juiste oplossing, maar niet fijn voor de docent om mee te werken;
- een berekening met variabelen waaraan geen waarden zijn toegekend (wordt dit gesuggered door BS 5.14 t/m 5.17 ?). Een foute oplossing;
- eerst een toekenning van waarden aan de variabelen, daarna een berekening met die variabelen.

De laatste oplossing verdient de voorkeur, i.v.m. de latere leesopdracht. De eerste oplossing kan mogelijk voorkomen worden door een aanwijzing van de docent.

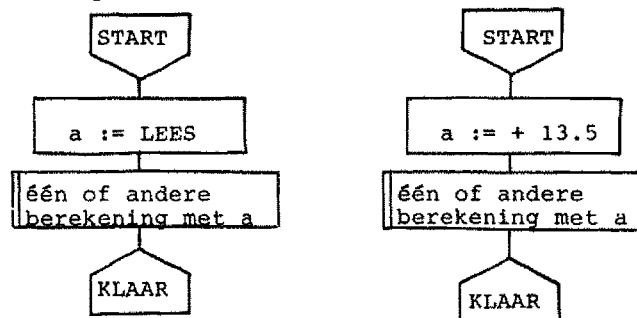
2.1.6 S_c_h_r_i_j_v_e_n__e_n__l_e_z_e_n (Compk hfdst 6, Wrks hfdst 3 en 9)

De schrijfpdracht geeft in het algemeen weinig problemen. Op het moment dat de programma's inderdaad verwerkt worden, kan nog enige aandacht worden geschonken aan het preciese effect van de opdracht SCHRIJF := a zoals: hoeveel cijfers voor en achter de decimale punt, wanneer wel en wanneer geen voortekenen, welke notatie bij een getal van een groot aantal cijfers voor de punt.

Het verwisselen van linker en rechter lid bij de lees- en de schrijfpdracht komt wel voor.

Toch is het principe gehandhaafd, dat rechts van het wordt-teken de bron staat en links het doel.

De leesopdracht.

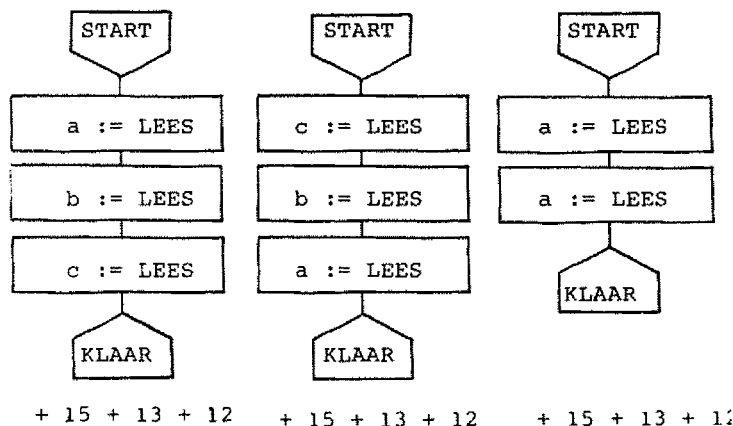


getalkaart + 13.5

Het voordeel van het linker blokschema boven het rechter is aan leerlingen niet zondermeer duidelijk. Dit voordeel moet verdedigd worden met het feit, dat het linker programma algemener bruikbaar is, d.w.z. dat de berekening uitgevoerd zou kunnen worden met verschillende waarden voor a. Dit wordt echter pas realiteit als het programma herhaald wordt door middel van een cyclus. Men lette overigens op het verschil tussen algebra en een computerprogramma. In de algebra wordt een berekening met letters uitgevoerd, zodat achteraf substitutie van waarden kan plaatsvinden, in een computerprogramma vindt eerst 'substitutie' plaats d.m.v. de leesopdracht en daarna de berekening met namen.

De nadruk moet gelegd worden op het feit, dat na KLAAR getallen genoemd worden *zonder* namen van geheugenplaatsen. Het blokschema moet zorgen dat deze getallen de juiste geheugenplaatsen gaan bezetten.

Oefening met eenvoudige gevallen is vaak wenselijk, zoals: welk getal zit er in geheugenplaats a in elk van de volgende gevallen?



2.1.7. De eerste programma's (Compk hfdst 7, Wrks hfdst 5)

Deze leerstof bevat de volgende nieuwe elementen:

- overgang van blokschema naar programma
- de uitvoeropdrachten TEKST en NR (dit geldt niet voor Wrks, waarin deze opdrachten in hfdst 4 behandeld worden)
- het opnemen van naam en klas in het programma
- de plaats van de te lezen getallen (geldt niet voor Wrks)
- het omzetten van het programma in een voor de computer acceptabele vorm zoals de schrapkaart).

Het is af te raden dit alles in één les te behandelen. Verdeling over meerdere uren zal wel met zich meebrengen, dat meer oefenstof zal moeten worden bedacht.

Aanwijzingen over het gebruik van de schrapkaarten kunt U vinden in §6.1. Bij eenvoudige (onvertakte) programma's lijkt een blokschema overbodig. Toch is het een goede gewoonte om elke verwerking door een computer in zes stappen te nemen:

- blokschema
- programma, geschreven op papier
- programma, geschreven op de schrapkaarten
- opdrachten aanschrappen op de schrapkaarten
- verzending
- bij terugontvangst, controle van de resultaten.

De eerste programma's dienen vooral om de theorie over geheugenplaatsen aan de werkelijkheid te toetsen.

Voorbeeld:

```
START
TEKST := "j.f,3a"
NR
a := 1
b := 3
SCHRIJF := a
SCHRIJF := b
NR
c := a
a := b
b := c
SCHRIJF := a
SCHRIJF := b
SCHRIJF := c
KLAAR
```

In sommige computers wordt geaccepteerd dat de opdracht $a := b$ wordt gebruikt, terwijl aan b nog helemaal geen waarde is toegekend. Wel blijkt a dan een volkomen absurde waarde te hebben.

De verklaring is de volgende:

Het ECOL-programma wordt gelezen, bekeken op onbekende hoofdletter-woorden e.d.. Elke onduidelijkheid in de ECOL-tekst geeft aanleiding tot foutmeldingen. De opdracht $a := b$ is op zichzelf niet fout en wordt daarom overgenomen. Tevens wordt aan elke voorkomende variabele een geheugenplaats toegewezen (voor insiders: real gedeclareerd). Totdat het programma de juiste waarde toekent aan die variabele is de inhoud van zo'n geheugenplaats een onvoorspelbare chaos. Als b in het programma geen waarde heeft gekregen, zal bij de opdracht $a := b$ de toevallig in b aanwezige rommel worden overgedragen aan geheugenplaats a .

Door de meeste docenten wordt de mening gekoesterd, dat juist in de eerste programma's zo weinig mogelijk door de leraar moet worden gecorrigeerd. Laat de leerling ervaren dat de computer een slecht verstaander is, die precies op de juiste wijze moet worden 'aangesproken'. Foutmeldingen zijn anders het onherroepelijke gevolg. Bij latere meer ingewikkelde programma's kan de docent foutjes tegen de taal vóór het verzenden laten verbeteren. Hebt U hierover een andere mening, dan voegt U zich bij een minderheid, die zeker niet klein is.

Probeer U de leerlingen te overtuigen, dat het meer zin heeft de eerste programma's zo te schrijven dat ze nagerekend kunnen worden. Pas dan kunnen we zeker zijn over de werking van de programma's. Hierdoor wordt voorkomen dat de leerlingen gaan wedijveren in het gebruik van immens grote getallen en enorme aantallen geheugenplaatsen. De resultaten van programma's die hierin uitblinken, hebben voor de leerling toch geen betekenis of controlebaarheid.

Programma's genoteerd op schrapkaarten zijn gemakkelijk te verbeteren en opnieuw in te zenden. Voorwaarde is wel dat de leesbare tekst bovenaan de kaarten staat aangegeven, zodat de onjuiste kaarten gemakkelijk kunnen worden teruggevonden en vernieuwd.

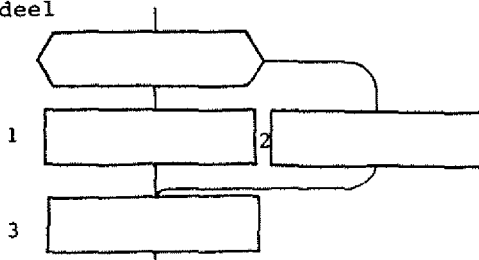
Bij de resultaten krijgt de leerling ook het programma nog afgedrukt, zodat gecontroleerd kan worden of dit via de schrapkaarten juist is doorgegaan.

2.1.8 V e r t a k k i n g i n b l o k - s c h e m a (Compk § 8.1, Wrks hfdst 7)

Een veel voorkomende fout is het gebruik van het wordt-teken in plaats van het gelijk-teken in het vraagblok. In Wrks hfdst 7 komt bovendien oefening voor met dezelfde geheugenplaatsnaam aan weerszijde van het wordt-teken en een eerste kennismaking met de tellende cyclus. Elke oefening hierin is broodnodig, alle vraagstukken worden daarom van harte aanbevolen.

2.1.9 D e s p r o n g o p d r a c h t e n (Compk § 8.1, Wrks hfdst 7)

Het blokschemadeel

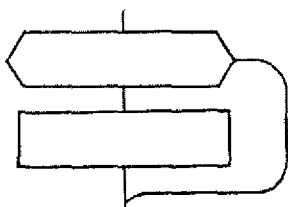


vereist bij omzetting naar een programma de beide sprongopdrachten:

```

ALS ..... DAN 1 ANDERS 2 (voorwaardelijke
1 ..... sprong)
    NAAR 3
2 .....
3 .....
  
```

U kunt ook eerst alleen de voorwaardelijke sprongopdracht introduceren door voorbeelden te zoeken, waarvan de splitsing de volgende gedaante heeft.

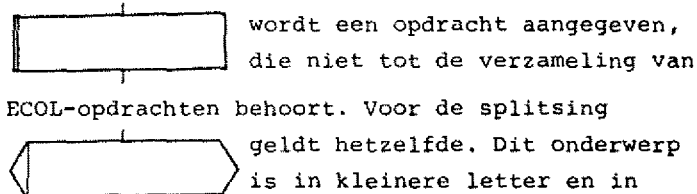


Veel voorkomende fouten bij het programmeren met splitsingen zijn:

- in de sprongopdracht staat een nummer genoemd dat niet als regelnummer voorkomt. De foutmelding verwijst meestal naar het ontbreken van een regelnummer voor KLAAR. Dat is namelijk de laatste regel, waarop het gezochte nummer zou hebben kunnen staan.
- een nummer wordt meerdere malen als regelnummer gebruikt
- een wordt-teken tussen ALS en DAN, in plaats van een gelijk-teken.

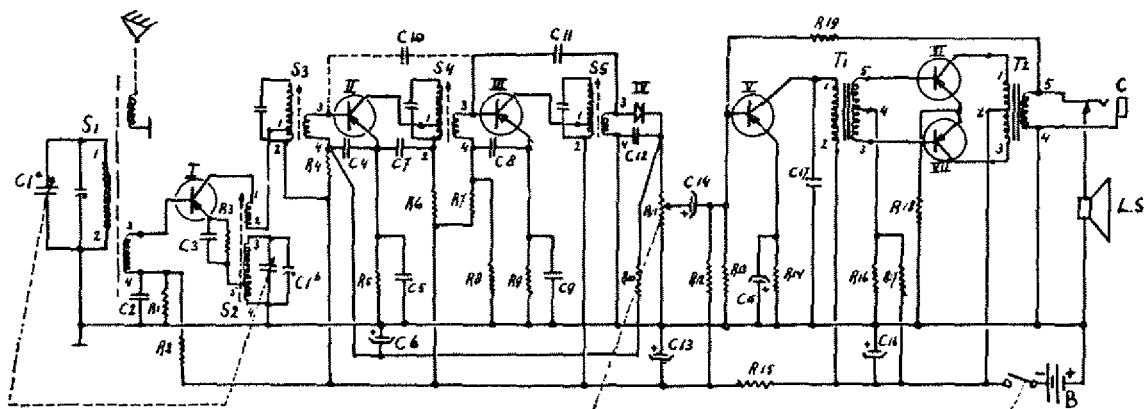
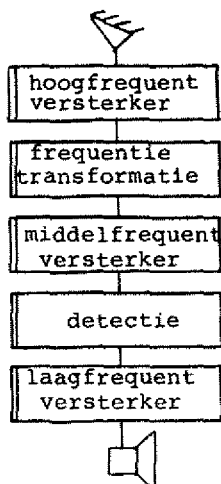
Het grove blokschema (Compk § 8.3, 8.4)

Bij het organiseren van een systeem is het zeer gebruikelijk om eerst een oplossing aan te geven in grote stappen, waarna elke stap in details uitgewerkt kan worden door verschillende mensen of teams. In de bedoelde paragrafen is dat aangegeven door zogenaamde grove blokken. In



Wrks helemaal niet opgenomen, omdat in het verleden is gebleken, dat door tijdgebrek het onderwerp vrijwel niet ter sprake kwam. Toch wil ik graag een lans breken voor de behandeling ervan. Naast tijdgebrek werd ook wel geringe belangstelling van de leerling als argument aangevoerd, maar juist de detaillistische instelling van leerlingen zou hiermee doorbroken kunnen worden. Ook het werken in teamverband kan bevorderd worden, omdat het grove blokschema mogelijkheid schept het probleem te verknippen.

In ieder geval zou ik U als docent willen aanbevelen om grove blokschema's te gebruiken om grote lijnen van oplossingen aan te geven. Men vergelijke dit bijvoorbeeld met de grove organisatie en een detailschema van een radio.

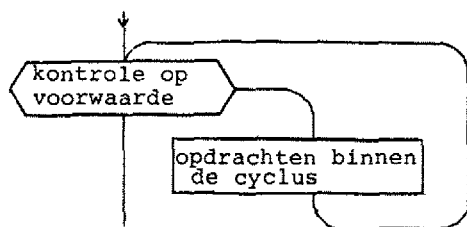


2.1.10 D_e_c_y_c_l_u_s

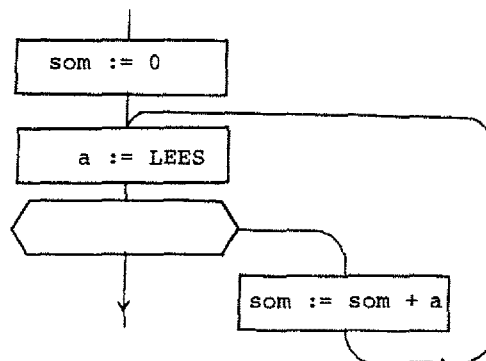
De grote kracht van de computer, het vele malen herhalen van een gedeelte van een rekenproces, waarbij deze herhaling aangegeven kan worden door de automaat terug te voeren naar een punt eerder in het programma. De cyclus maakt het mogelijk veel werk te laten verzetten met weinig programmeren. Onze leerlingen kunnen bij dit onderwerp weer een algoritmische ervaring opdoen: het zoeken naar struktuurdelen die zich herhalen. Overigens moeten sommige leerlingen hiertoe wel worden aangezet. Maar al te vaak verkiezen ze het om een programmadeel vijfmaal uit te schrijven, boven het construeren van een cyclus. Dit laatste vereist namelijk meer denkwerk. De cyclus eindigt op een zekere voorwaarde. Dit kan zijn het ontmoeten van een bijzonder getal op de getalkaarten of het bereiken van een zekere waarde door een variabele. Vergeet men de controle op het voldoen aan deze voorwaarde in te bouwen, dan spreken we van een oneindige cyclus. Een oneindige cyclus zou een computer te lang bezet houden, vandaar dat een veiligheid in het systeem is ingebouwd, die de programmaverwerking kan beëindigen na een bepaalde tijd (bijv. 15 sec., afhankelijk van de snelheid van de machine) of na een bepaald aantal sprongen (500 of 1000) veroorzaakt door sprongopdrachten.

Dit afsnijden van de verwerking (de guillotine) kan ten onrechte zijn, in het geval de berekening eindig doch tijdrovend is. In dat geval kan een ruimere tijd worden aangevraagd voor dat speciale programma.

Voor de cyclus geldt (uitgezonderd de tellende cyclus) hetzelfde als wat in 2.1.2 is gezegd, nl. dat een oplossing van de volgende vorm meestal de enig juiste is

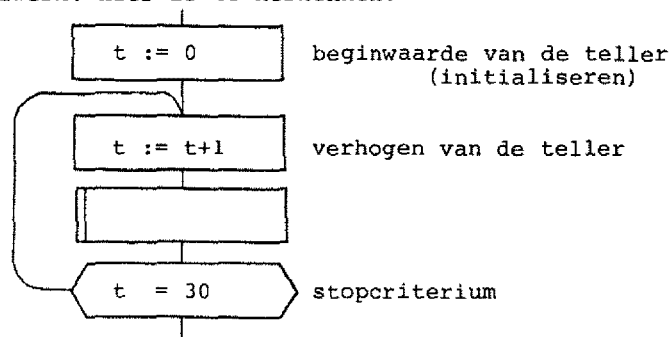


Veel opgaven (Compk 9.1 t/m 9.4, 9.12, 9.17 t/m 9.23, Wrks 9-17, 9-18, 9-19, 10-3, 10-4, 10-5, 10-6) hebben betrekking op een getallenrij, die aan de programmaschrijver onbekend is. De schrapkaarten geven de mogelijkheid om dit realiteit te laten worden, want U kunt een leerling opdragen het programma te schrijven en zijn buurman om de getalkaarten te leveren.



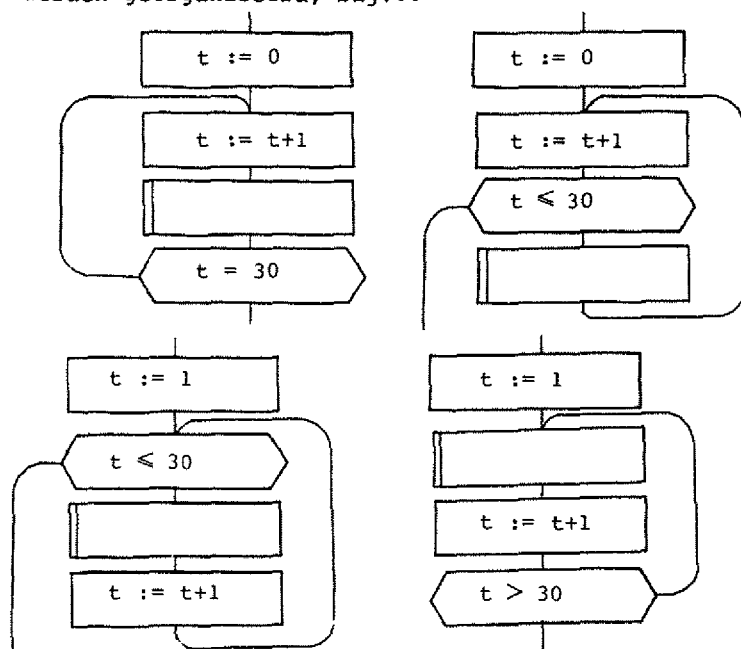
In de constructie, waarin de geheugenplaats "som" als spaarpot fungeert voor het sommeren van een aantal getallen a, moet uitdrukkelijk gewezen worden op de bovenste opdracht (het leeggooien van de spaarpot).

Een bijzonder geval van de cyclus is de tellende cyclus, waarin de voorwaarde voor beëindiging (stopcriterium) gekonstrueerd wordt door een soort telwerk. Hier is te herkennen:



Bij door leerlingen ingezonden programma's mankeert het nogal eens aan één van deze drie noodzakelijke opdrachten. Als docent kunt U vóór het inzenden hier misschien enige controle op uitoefenen.

De tellende cyclus kan op verschillende manieren worden georganiseerd, bijv.:



Het verdient de voorkeur om als docent U zoveel mogelijk in de voorbeelden te houden aan één type.

2.1.11 B e p a l e n v a n g r o o t s t e o f k l e i n s t e g e t a l (Compk § 9.5, Wrks 9-21, 9-22)

De algoritme voor het bepalen van het grootste (of kleinste) van een aantal op kaarten staande getallen behoort voor veel leerlingen niet tot de gemakkelijkste. Hier volgt een verslag van een VWO-klassegesprek, zodat U zich alvast op het onderwerp kunt oriënteren.

docent

+ 76 + 78 + 77 + 79 + ... + 76 + 0.

Dit stelt een serie resultaten voor van een wedstrijd speerwerpen, behalve natuurlijk de afsluiter nul. Het grootste getal uit deze rij zullen we rekord noemen. Denk je in dat de rij zo'n vijftig getallen lang is. Breng eens onder woorden hoe je zelf het rekord zou opzoeken.

leerling

Je kijkt of je een getal groter dan 79 tegenkomt.

docent

Ja, wij mensen kunnen in één oogopslag het grootste van de eerste vier getallen zien. Dit grootste getal onthouden we en we gaan de rij langs om een groter getal te zoeken. Stel dat we het getal 80 tegenkomen, wat dan?

leerling

Dan gaan we kijken of we een getal groter dan 80 zien.

docent

Is het getal 79 nog belangrijk?

klas

nee.

docent

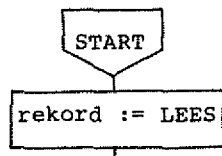
Wanneer zou je zeker weten dat 80 het rekord is?

leerling

Als je bij de afsluiter komt zonder een getal groter dan 80 gezien te hebben.

docent

Juist. We gaan eens kijken of we dit werkje door een computer kunnen laten doen. De computer kan niet in één oogopslag vier getallen overzien. We lezen één getal



leerling

Maar het eerste getal is helemaal het rekord niet.

docent

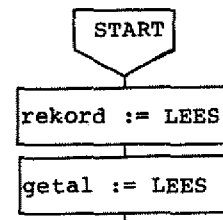
In ons voorbeeld niet, maar kun je ook een geval bedenken waarbij dat wél het geval is?

leerling

Als de volgende getallen toevallig allemaal kleiner zijn dan het eerste getal.

docent

Zolang we nog niet verder gekeken hebben, nemen we aan dat het eerste getal het grootste is. We gaan nu het volgende getal uit de rij lezen:



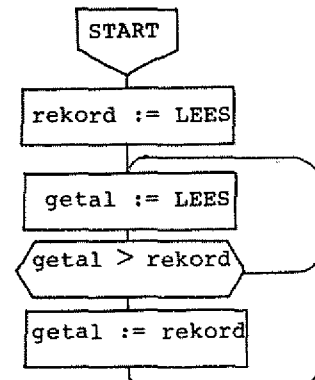
Wat doen we met dit getal?

leerling

Kijken of dit groter is dan het rekord wat we al hebben en als dat zo is, wordt dat het nieuwe rekord.

docent

Goed. Ik zal nu een blokschema tekenen, waarin een aantal fouten zitten. Aan jullie om ze eruit te halen.

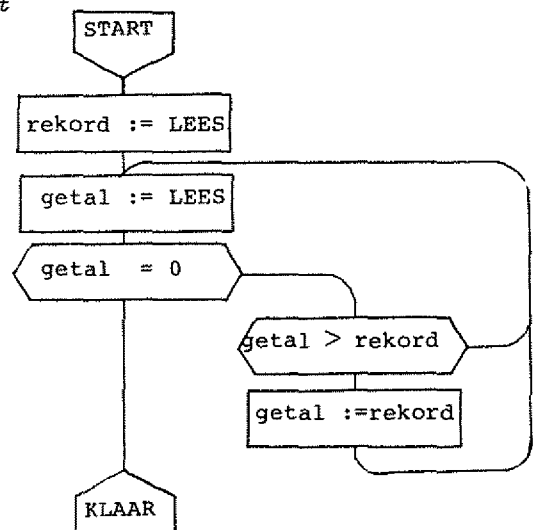


fout
blok-
schema

leerling

Het is een eindeloze cyclus. We moeten nog kijken of we de nul tegenkomen.

docent



fout
blok-
schema

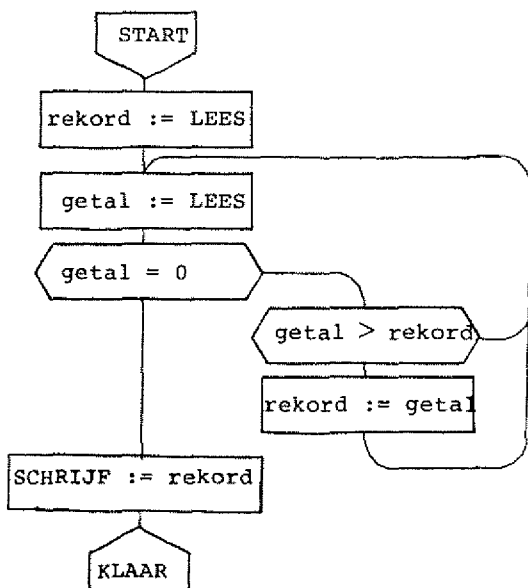
leerling

Er moet nog een schrijfpdracht in.

leerling

Dat getal wordt rekord moet toch andersom?

docent



+ 76 + 78 + 77 + 79 + 77 + 80 + 81 + 76 + 0

Jan, maak op het bord eens een geheugenschema om dit blokschema met deze getallen te controleren.

rekord	getal
+ 76	
+ 76	+ 78
+ 78	+ 78
+ 78	+ 77
+ 78	+ 79
enz.	

docent

Je kunt je voorstellen dat bij wedstrijden alle uitslagen door een computer verwerkt worden en dat de wedstrijd speerwerpen is afgelast. Op de getalkaarten staat dan alleen + 0.

Werkt het blokschema nu goed?

klasa

Nee. Het blokschema wil direkt al twee getallen lezen.

leerling

(Maakt met behulp van vraagblok na rekord := LEES ook hiervoor een voorziening).

docent

Nu staat er op getalkaarten:

+ 10.41 + 10.52 + 10.35 + 10.40 + 0
en dat zijn de tijden gemaakt in de wedstrijd
hardlopen over 100 m. Wat is nu het rekord?

klas

Het kleinste getal.

docent

Wat moet gewijzigd worden in het blokschema om zo'n kleinste getal te vinden?

klasa

(na enig zwijgen). Alleen het groter-teken moet een kleiner-teken worden.

docent

Zijn er nog opmerkingen?

leerling

Wat gebeurt er als zo'n grootste getal (of kleinste getal) twee keer voorkomt?

docent

Dan is het rekord blijkbaar geëvenaard, maar dat verandert niets aan het rekord. Probeer het blok-schema maar eens met de getallen $+76 + 80 + 81 + 77 + 81 + 0$.

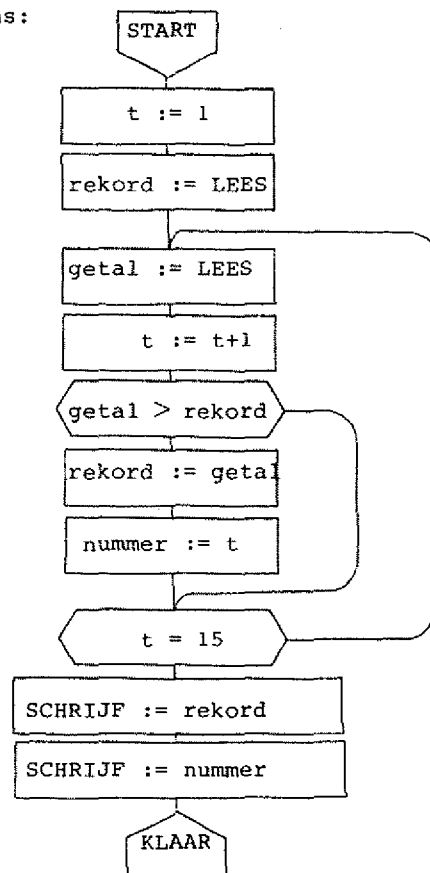
leerling

Je weet nu wel wat het rekord is, maar niet wie het behaald heeft.

docent

Dat is waar. We zullen de volgende keer een opgave maken waarbij we de rugnummers zullen betrekken, zodat we ook het rugnummer van de rekordhouder weten. (Compk opgave 9.22). Verdelen jullie je nu in groepjes van twee. Eén schrijft een programma voor het vinden van het grootste, de ander voor het vinden van de kleinste van een aantal getallen. Schrijven jullie voor elkaar de getalkaarten en onthoudt welk antwoord er volgende week bij de ander uit moet komen.

Men lette er bij Compk opgave 9.22 op dat een programma volgens:



een onjuiste oplossing biedt voor het geval het eerste getal het grootste getal is.

Het programma van Compk opgave 9.23 mag niet beginnen met

```
START
grootste := LEES
kleinste := LEES
```

gezien bijvoorbeeld de getallenrij
+ 1 + 3 + 2 + 2 + 2 + 0.

2.1.12 D_e____r_i_j

(Compk hfdst 10, Wrks hfdst 11)

De rij wordt door veel leerlingen een moeilijk onderwerp gevonden. Dit heeft ondermeer de volgende oorzaken:

- het nut van de rij is niet duidelijk
- er zijn bijna geen eenvoudige opgaven, waarin de rij onmisbaar is.

Het nut van de rij wordt duidelijk gedemonstreerd aan de volgende opgave (ongeveer Compk opg. 10.1)

Op getalkaarten staan 100 getallen. Laat een lijst schrijven waarin staat aangegeven hoever elk getal van het gemiddelde van die 100 getallen afligt (onder hoever verstaan we nu maar even getal min gemiddelde zonder de absolute waarde te nemen).

- Elk van de honderd getallen toekennen aan variabelen a,b,c,d enz. enz. is mogelijk. Dit zou alleen nut hebben als de variabelen verschillende bewerkingen zouden ondergaan. Het is omslachtig om aan te geven: doe met variabele c hetzelfde als met a.
- Elk van de honderd getallen toekennen aan dezelfde variabele is mogelijk voor het berekenen van het gemiddelde.

```
START
t := 0
som := 0
1 t := t+1
a := LEES
som := som+a
ALS t<100 DAN 1
gem := som/100
KLAAR
```

Na deze berekening is de getallenrij niet meer beschikbaar. De opgave kan zo niet worden opgelost.

- De rij is vrijwel noodzakelijk als men een aantal variabelen eenzelfde behandeling wil laten ondergaan, waarna de waarden nog steeds beschikbaar moeten zijn.

```
START
RIJ(1:100)a
t := 0
som := 0
1 t := t+1
a(t) := LEES
som := som+a(t)
ALS t<100 DAN 1
gem := som/100
t := 0
2 t := t+1
SCHRIJF := a(t)-gem
ALS t<100 DAN 2
KLAAR
```

Eenvoudige opgaven waarin een rij gebruikt moet worden zijn helaas zeldzaam. Opgaven zoals § 2.3 nummer 45 zijn als oefening bruikbaar, maar weinig overtuigend. De nummers 42 en 44 zijn moeilijker maar daarentegen schoolvoorbeelden van nuttig rijgebruik.

Het komt voor dat leerlingen, nadat de rij behandeld is, te onpas gebruik van rijen maken. Bijvoorbeeld elke verzameling op getallenkaarten wordt, ongeacht de bewerking die zij moet ondergaan, opgeslagen in een rij. Hiertegen zou een waarschuwend geluid op zijn plaats zijn. Het verdient aanbeveling om bij minder begaafde leerlingen de rij vrijwel alleen ten tonele te voeren in verband met bestanden.

2.1.13 B_e_s_t_a_n_d_e_n

(Compk § 13.3, Wrks hfdst 11)

Het achtergrondgeheugen biedt de mogelijkheid om gegevens op te slaan voor langer dan alleen de tijd dat het programma werkt. Het zelf opslaan van gegevens om daarover na één of meer weken weer te kunnen beschikken, is voor leerlingen niet van groot belang, hoewel niet onmogelijk. Het grote voordeel in de les is dat de leerling zijn programma's kan laten opereren op gegevens die hem werkelijk onbekend zijn maar onmiddellijk ter beschikking staan. De programma's winnen aan realisme voor de leerling wanneer hij niet zelf of zijn buurman de gegevens in de vorm van getalkaarten aan het programma hoeft toe te voegen.

Een probleem voor veel leerlingen is het uit elkaar houden van de opdrachten rijnaam := BESTAND(n) en BESTAND(n) := rijnaam.

Als docent kunt U in Uw voorbeelden deze opdrachten bijvoorbeeld uitspreken als: (de rij) rijnaam := (gevuld met de getallen uit) BESTAND(n) en BESTAND(n) := (gevuld met de getallen uit de rij) rijnaam.

Bij onderwerpen als girodienst, vliegtuigreservering (Compk § 15) en personeelsadministratie komt het gebruik van bestanden op elementaire wijze

naar voren. Van het laatste onderwerp tenslotte nog een klassegesprek.

De docent zorgt dat de leerboekjes gesloten blijven,

docent

We gaan een administratie opzetten voor een bedrijf met 200 werknemers. Dat wil zeggen dat we van elke werknemer bepaalde gegevens gaan onthouden. Natuurlijk willen we van elke werknemer weten wat zijn naam en adres is. Die staan op een lijst, genummerd van 1 tot 200. Wat moet er nog meer bekend zijn op het bedrijf van elk van deze mensen?

leerling

De salarissen.

docent

Is dat nodig?

leerling

Er moet toch bekend zijn wat er aan elk moet worden uitbetaald aan het eind van de maand.

docent

Juist. Wat nog meer?

leerling

De leeftijden.

docent

Is dat nodig?

leerling

Je moet weten wanneer iemand met pensioen gaat.

docent

Goed. Wat nog meer?

leerling

Of het een man of een vrouw is.

docent

Ik kan me voorstellen, dat je dat zelf graag wilt weten, maar is het ook nodig voor onze administratie?

leerling

Misschien niet.

docent

Iets anders dan.

leerling

Of hij getrouwd is, ja of nee.

docent

Waarom dat dan?

leerling

Dat scheelt in de belasting.

docent

En wat heeft onze administratie met die belasting te maken?

leerling

Die wordt toch van het salaris afgetrokken?

docent

Juist, de loonbelasting wordt op het salaris ingehouden.

leerling

Maar een vrouw betaalt toch minder belasting dan een man.

docent

Een werkende gehuwde vrouw betaalt inderdaad minder, wat wil je met deze opmerking?

leerling

Dan moet de administratie ook weten of het een man of een vrouw is.

docent

Goed, we noteren dus ook het geslacht. Even verza-
melen: behalve naam en adres administreren we sa-
laris, leeftijd, geslacht, al of niet gehuwd zijn.
Iemand nog meer?

leerling

Hoelang hij bij de firma werkt.

docent

Is dat nodig?

leerling

Voor jubilea.

docent

Goed, dat er dus ook bij. Je hebt wel gemerkt dat ik telkens vroeg: "is dat nodig?". Als je een administratie gaat opbouwen, moet je je eerst afvragen welke gegevens je nodig hebt. Daarna hoe we deze gegevens zullen opbergen. De 200 salarissen brengen we onder in bestand 19. Laten we aannemen dat ze afgerond zijn op gehele guldens, bijvoorbeeld +1071+1280+1368+....enz. Waarom is het niet erg handig om op dezelfde manier de leeftijden op te bergen?

leerling

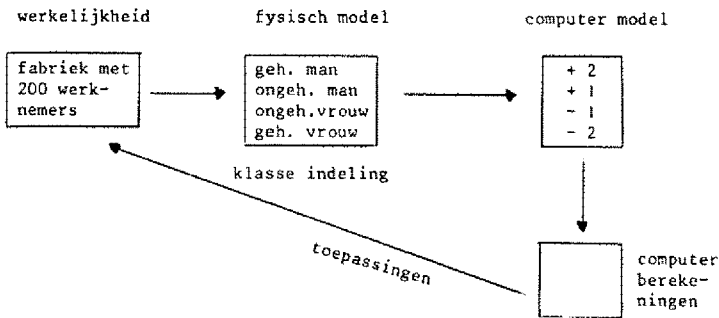
Als we de geboortejaren noteren, dan hoeven we niet meer te wijzigen. De leeftijden veranderen steeds.

docent

Goed zo. Je ziet, ook de manier van opbergen van gegevens kan meer of minder handig. Nu het geslacht en het gehuwd zijn. Dit zijn gegevens die niets met getallen te maken hebben. Toch kunnen we in ons bestand alleen getallen opnemen. Hoe lossen we dat op?

We verlaten hier de klas. De rest van het gesprek laat zich raden. Men lette op de laatste alinea waarin een representatieprobleem ter sprake komt. Niet numerieke gegevens moeten door getallen worden gerepresenteerd. Bij de behandeling van "grote" computertoepassingen komt het maken van een model van de werkelijkheid duidelijk naar voren. Men kan geen berekening maken aan de werkelijkheid zelf, maar de werkelijkheid moet gerepresenteerd worden door een computermodel en aan dit model worden de berekeningen gemaakt. Naast een voorbereiding op de maatschappij waarin computertoepas-

singen niet meer zijn weg te denken en naast het leren relativiseren van de uit deze toepassingen verkregen resultaten (deze zijn niet beter dan het model waaraan de berekeningen plaats vonden) is dit proces een eenvoudig en zeer illustratief voorbeeld van wat men in de wiskunde onder "matimatiseren" verstaat.



```

START
a := LEESY
b := 10
SCHRIJFSYM := a
NR
TEKST := "a"
NR
SCHRIJF := a
NR
SCHRIJFSYM := b
NR
TEKST := "10"
NR
SCHRIJFSYM := b+30
NR
TEKST := "b+30"
NR
SCHRIJFSYM := a+b
NR
SCHRIJF := a-b
NR
TEKST := "a-b"
KLAAR
b
  
```

In Compk opgave 14.1 en enkele volgende is niet exact te bepalen welke leeftijd de werknemers hebben uit het bestand met de geboortejaren. Dat hangt immers van de datum af, waarop de peiling plaats vindt. Laat U Uw leerlingen dit zelf ontdekken. Alleen een feest op oudejaarsavond (Wrks 12-7) geeft zekerheid omtrent de leeftijden.

De invoering van de schrapkaart heeft figuur 16.2 minder van toepassing gemaakt. Vraagstuk 16.4 levert ook daardoor problemen. Het symbool $\geq$ aangestreept op de schrapkaart wordt vertaald in twee tekens $>=$, met achtereenvolgens de interne waarden 86 en 84. Symboolwaarde 88 kan dus (nog) niet via een symbool op een schrapkaart worden ingevoerd. SCHRIJFSYM := 88 heeft echter wel degelijk het afdrucken van $\geq$ ten gevolge.

2.1.14 Manipuleren met letters (Compk §16)

Een gewild onderwerp bij veel leerlingen, omdat het speelse elementen combineert met een groot aantal toepassingen.

Op twee toepassingen wordt iets nader ingegaan. Bij de eerste behandeling van dit onderwerp ontstaan bij leerlingen nog weleens misverstanden over de interne representatie, bijvoorbeeld de volgende.

- De symbolen staan zelf in de geheugenplaatsen.
- De computer herkent de getalrepresentatie als afkomstig van een letter, je mag er vast niet zomaar mee rekenen.

Oefeningen als de volgende kunnen dit snel oplossen. Wat wordt achtereenvolgens afgedrukt door het volgende programma:

Naar aanleiding van opgave 16.11 kan ter sprake komen dat het zetten van kolommen veelal door computers wordt geleid. Opgave 16.12 gaat hierop dieper in. In werkelijkheid zal men op het eind van de regel ook het afbreken van woorden toelaten. Men kent dan ook programma's die zonodig nederlandse woorden kunnen afbreken. Het voert te ver om dergelijke programma's te behandelen. Wel kan gewezen worden op de beperkingen van deze programma's. Het is al heel moeilijk om een computer te laten bepalen hoe "kwartslagen" moet worden afgebroken: het zal van de context afhangen of "kwartslagen" of "kwarts-lagen" bedoeld werd. Leerlingen kunnen gemakkelijk in kranten de fouten opsporen ten gevolge van het automatisch kolommen zetten.

Voorbeeld:

...nog andere vliegende
vleermuizen en vogels. Onder de reptielen zijn er een paar die ruim honderd miljoen jaar geleden vleugels hebben gekregen. Bij die vliegende reptielen is slechts de ringvinger sterk verlengd en de vlieghuid is slechts daaraan bevestigd. Duim, wijs- en middelvinger blijken kleine stompjes met sterk gekromde nagels te zijn. De pink is bij het

Bij het volgende is duidelijk te zien dat het programma er vanuit gaat dat "on" altijd een lettergreep is.

Iets dergelijks treft men bij de grote uitgestorven reptielen aan: geweldig ontwikkelde achterpoten, maar voorpotjes die alleen nog maar kleine, onbetekenende aanhangsels zijn. Dit

Naar aanleiding van §16 kan ter sprake komen het gebruik van computers in de taalwetenschappen. Hierover het volgende.

Onder klassifikatieprocessen verstaan wij het omwerken van een tekst in een gegeven orde (lopende tekst, woordenboek of frekwentielijst bv.) tot een andere orde. Dit is voor de computer de meest elementaire bewerking.

Indexen en Concordanties

Een index is de (meestal) alfabetische ordening van de woorden van een tekst voorzien van de plaats(en) die ze in die tekst hebben. Een concordantie verschaft niet enkel vindplaatsen maar geeft ook een hoeveelheid context. Deze varieert naar gelang van het programma dat de computer instrueert. In de *Woordindex op Mei van Gorter* vinden wij de woorden alfabetisch geordend met opgave van de bladzijde, de versregel en het nummer van het woord in de regel. Zo vinden wij *koel* aldus opgegeven:

Koel (frekw. = 8) 19-16-5; 33-30-6; 82-26-6;
101-22-1; 124-06-1; 155-07-2;
155-07-7; 156-09-9.

Bij een concordantie waarbij de lengte van de context zou begrensd worden door de twee meest nabije zintekens waartussen het woord staat, zouden we verkrijgen:

koel 19-16-5, en laat zijn nog zijn voeten koel in dauw van wei.
33-30-6 - koel lag nog de wel -
82-26-6 - een hinde gelijkend draafde ze uit een koel woud.
101-22-1 ? Koel is de maan, enz.

Door dit voorbeeld zal het nut van indexen en concordanties wellicht duidelijk(er) geworden zijn; onderzoek naar de combinatie- en associatiemogelijkheden

van een woord (zijn semantisch-stilistisch veld), naar temata, motieven en beeldspraak bij een auteur, syntaktisch onderzoek (zoals het gebruik van pronomiale verwijzingen of de combinatiemogelijkheden van structurele woorden) e.d. kunnen nu meer dan ooit op een betrouwbare, want exhaustieve wijze onderzocht worden.

Frekwentielijsten

De computer kan van om het even welke tekst niet alleen een index of een concordantie maar ook een frekwentielijst maken. Het volstaat dat hij de woorden alfabetisch rangschikt en identieke woorden samentelt.

Dit kan dan het uitgangspunt worden voor verder onderzoek. Bij het vervaardigen van de frekwentielijst op *Mei* ontdekten wij bv. dat het woord *koel* (+ *koele*, *koelen* (a.), *koeler*) 17 maal voorkwam. Daarnaast kwamen ook nog verwante woorden voor als *koelende* (1x), *koelte* (4x), *koelvat* (1x). Een woord als *fris*(ser) daarentegen kwam slechts éénmaal voor.

Het zou interessant zijn deze frekwentie te kunnen vergelijken met de frekwentie die *koel* en *fris* in een algemene frekwentielijst voor het Nederlands zouden bezitten.

Een grote algemene frekwentietelling van Nederlandse woorden en structuren zou dus niet allen bij didactisch, maar ook bij zuiver linguïstisch en literair onderzoek zeer waardevol zijn juist door het preciezer inzicht in normen en normafwijkingen dat hieruit zou kunnen gewonnen worden. Een dergelijke lijst blijft echter ondanks de mogelijke computerhulp nog steeds uit.

Retrograde lijsten

Bij een retrograde lijst worden de woorden alfabetisch gerangschikt naar de eindletter i.p.v. naar de beginletter zoals gebruikelijk. Men bekomt aldus een lijst die voor het Nederlands bv. in het begin worden zou bevatten als: *tuba*, *logica* en *mica* en op het einde *ersatz*, *hertz* en *jazz*. Verder zouden woorden eindigend op *-man*, *-ig*, *-heid*, *-tuig*, *-ing*, enz. allen samengegroepeerd zijn. Het nut van dergelijke lijsten ligt dus vooral op het gebied van de *woordvorming*. Door retrograde rangschikking immers komen woorden met hetzelfde eindmorfeem samen.

Vergelijking inzake suffixen tussen talen onderling (met eventueel onderzoek van interferentieverschijnselen zoals het gebruik in het Nederlands van *-atie* bij woorden waar het Frans geen *-ation* kent) is mogelijk, aangezien voor de meeste (althans Westerse) kultuurtalen reeds een dergelijk instrument bestaat. Verder kan een retrograde lijst ook dienst bewijzen als rijmwoordenboek en als hulp bij reconstructie van teksten.

Waar, zoals gezegd, voor de meeste talen *retrograde woordenboeken* bestaan, zijn *retrograde frekwentielijsten* nog vrij zeldzaam. Daarmee bedoelen wij de retrograde rangschikking van een algemene frekwentielijst. Dit schijnt ons nochtans een waardevolle aanvulling te zijn op het retrograde woordenboek. Aan de hand van een retrograde frekwentielijst kan men immers een probabilistisch model van taalgebruik voor suffixen of andere eindmorfemen opstellen waartegenover individuele "performances" kunnen worden afgewogen. Zo konden wij konstateren dat de diminutievorming bij Gorter (Mei) afweek van het gewone Nederlands: waar in de retrograde (algemene) frekwentielijst geen verkleinwoorden op *-ke* voorkwamen troffen wij bij Gorter: *kin-deke* (f=1), *maankindeke* (f=1), *windeke* (f=2), *bijke* (f=1), *woordekens* (f=1) en *zusterke* (f=1) aan.

2.1.15 Sorteren en zoeken (Compk §17)

Een wat moeilijker onderwerp dat echter een indicatie geeft voor efficiëntieproblemen bij computertoepassingen. Bij de girodienst kwam in 13.5 al de noodzaak van sorteren naar voren.

Het dagelijks sorteren van rijen ter lengte van enkele miljoenen getallen stelt hoge eisen aan de werkingssnelheid van de algoritme. Het verdient misschien aanbeveling ook andere algoritmen voor sorteren te behandelen.

In 17.1 werd beschreven het sorteren door het naar de n-de plaats schuiven van het grootste van n getallen, n loopt van het oorspronkelijke aantal terug tot 2. Een andere sorteermethode verwisselt steeds (zo nodig) twee na elkaar komende getallen.

15	13	13	13	13	13	7	7
13	15	15	15	7	7	13	12
49	49	7	7	15	12	12	13
7	7	49	12	12	15	15	15
12	12	12	49	49	49	49	49

Deze methode is iets onvoordeliger, omdat na het vergelijken van twee getallen er een verwisseling (3 opdrachten) kan plaats vinden, terwijl bij het zoeken naar de grootste van een rij getallen na het vergelijken slechts een vervanging (1 opdracht) noodzakelijk kan zijn. Het programma voor de hier beschreven methode heeft een redelijk eenvoudig uiterlijk.

```

.....
.....
MET -1, j := n, 2
MET 1, i := 1, j-1
ALS a(i) ≤ a(i+1) DAN 26
p := a(i)
a(i) := a(i+1)
a(i+1) := p
26 HERHAAL
27 .....

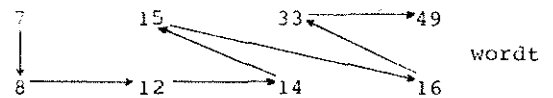
```

← vlag := 0
← vlag := 1
← ALS vlag = 0 DAN 27

Men kan bovendien een verfraaiing aanbrengen (de opdrachten met vlag) waardoor de algoritme stopt op het moment dat geen verwisselingen meer plaatsvinden (vlag blijft 0).

Er wordt geen overbodig werk verzet in een rij getallen die al op de juiste wijze gesorteerd blijkt te zijn. Ook hierbij ligt het aantal uitgevoerde opdrachten in de orde van $2\frac{1}{2}n^2$.

Een in de praktijk gebruikte algoritme is ingewikkelder. Men gaat hierbij uit van de gedachte dat het aantal opdrachten om twee gesorteerde rijen met lengte $\frac{1}{2}n$ in elkaar te schuiven tot een gesorteerde rij met lengte n van de orde n is.



7 8 12 14 15 16 33 49

Van elk getal kan men de opvolger bepalen door één opdracht, nl. het vergelijken van zijn oorspronkelijke opvolger en het kleinste getal uit de andere rij dat nog niet aan de beurt geweest is.

Het netjes in een rij zetten en het bijhouden van een kleine administratie vereist nog wel enkele opdrachten, maar gaat in het totaal toch niet 8 te boven.

Als men uitgaat van $n = 2^k$ te sorteren getallen, dan kan men deze rij opdelen in 2^{k-1} tweetallen en al deze tweetallen sorteren ten koste van $2^{k-1} \times 8 \times 2 = 8n$ opdrachten. Deze 2^{k-1} rijtjes kunnen twee aan twee in elkaar geschoven worden tot 2^{k-2} gesorteerde viertallen, ten koste van $2^{k-2} \times 8 \times 4 = 8n$ opdrachten weer twee aan twee in elkaar schuiven tot 2^{k-3} gesorteerde achttallen ten koste van $2^{k-3} \times 8 \times 2^3 = 8n$ opdrachten. Dit behoeft hoogstens k maal herhaald te worden om tot een gesorteerd n-tal te komen, d.w.z. een volledig gesorteerde rij.

In totaal verbruikt men $8nk$ opdrachten. Het op deze wijze sorteren van een rij ter lengte van een willekeurige n kost in het algemeen in de orde van $8n \cdot \log n$ opdrachten, wat voor grote n aanzienlijk vlugger gaat dan een algoritme van de orde n^2 . Vergelijk verder Compk §17.3.

4.1.16 S_i_m_u_l_a_t_i_e

(Compk §18.3, Wrks hfdst 14)

Een mijns inziens belangrijk onderwerp in het geheel van de computerkunde, omdat hierbij de modelvorming duidelijk naar voren komt. Als voorbeeld Wrks 14-8.

Iemand wil een uitkijktoren bouwen. Er komt een lift die elke minuut éénmaal op en neer gaat en waarin ten hoogste 20 mensen kunnen plaatsnemen. Simuleer de situatie door het aantal mensen in de lift bij elke tocht (naar boven zowel als naar beneden) te gokken, met de volgende uitzondering. De eerste drie minuten gaat de lift leeg naar beneden.

Zet de lift een uur en bedrijf en laat het grootste aantal mensen bovenin de toren bepalen. Hieruit zou de bouwer van de toren enigszins kunnen afleiden voor hoeveel mensen er ruimte moet zijn op de bovenste etage.

Denk eraan dat de lift nooit meer mensen naar beneden kan halen dan er boven aanwezig zijn.

```

START
max := GOK(0,20) + GOK(0,20) + GOK(0,20)
aant := max
MET 1, k := 4,60
aant := aant + GOK(0,20)
ALS aant > max DAN 1 ANDERS 2
1 max := aant
2 aant := aant - GOK(0,20)
ALS aant < 0 DAN 3 ANDERS 4
3 aant := 0
4 HERHAAL
TEKST := "het grootste aantal mensen in de
SCHRIJF := max                toren is"
KLAAR

```

De noodzaak van simuleren volgt uit het feit dat de uitkijktoren nog niet aanwezig is en er dus geen mogelijkheid is om uit de werkelijkheid een konklusie te trekken. De beperkingen van het model worden in de opgave al gesuggereerd door 'enigszins kunnen afleiden' enz. Het is immers niet gezegd dat men bovenin de toren beslist ruimte moet hebben voor het maximum aantal te verwachten mensen. Hierbij zullen de financiën ook een rol spelen. Vrijwel elke leerling zal aanvoelen dat de simulatie niet voorziet in een heldere zonnige vakantiedag waarop de lift zo lang mogelijk vol naar boven zal gaan.

De eerste drie keer gaat de lift leeg naar beneden omdat we aannemen dat iedereen tenminste 3 minuten van het uitzicht wil genieten alvorens

weer in de lift te stappen. Maar het bovenstaande programma houdt gedurende de rest van het uur hiermee geen rekening. Het kan voorkomen dat halverwege het uur het bovenin de toren leeg is en dat de eerstvolgende liftlading die boven komt grotendeels onmiddellijk weer vertrekt. Ook dat is allerminst realistisch. Een betere simulatie zou zijn om van elk bovengekomen individu het genot van het uitzicht te gokken tussen bijvoorbeeld 3 en 15 minuten. Hierbij moet dan weer rekening worden gehouden met de beperking tot 20 personen van de neergaande lift. Het uitprogrammeren van dit soort verbeteringen lijkt me veel minder belangrijk dat een goede discussie in de klas over de bezwaren tegen het model en de richting waarin verbeteringen mogelijk zijn.

De opgaven Compk 18.9 en 18.10 geven aanleiding tot wachttijd problemen. De betreffende programma's zijn beslist minder eenvoudig te noemen. Als voorbeeld van een uitgewerkt vraagstuk het volgende. (Cursus Computerkunde voor Mavo- en LBO-docenten, opgave 19).

Bij een automatische autowasinrichting gebruikt men een lijn die vier minuten voor het schoonmaken van een auto nodig heeft. Men weet uit ervaring dat de tijd tussen de aankomst van een klant en de aankomst van de volgende een toevalsgetal is tussen 1 en 6 minuten. Een rij van wel meer dan tien wachtende auto's komt nogal eens voor. Het bedrijf overweegt de installatie van nieuwe apparatuur waardoor de doorvoersnelheid verhoogd wordt tot maximaal één auto per drie minuten. Simuleer een werkdag van dit vernieuwde systeem en laat het grootste aantal wachtende auto's afdrukken. Het bedrijf opent om 8.30 en sluit om 18.00 uur.

De oplossing van dit vraagstuk vindt U op de volgende bladzijde in blokschema. Hierin is:

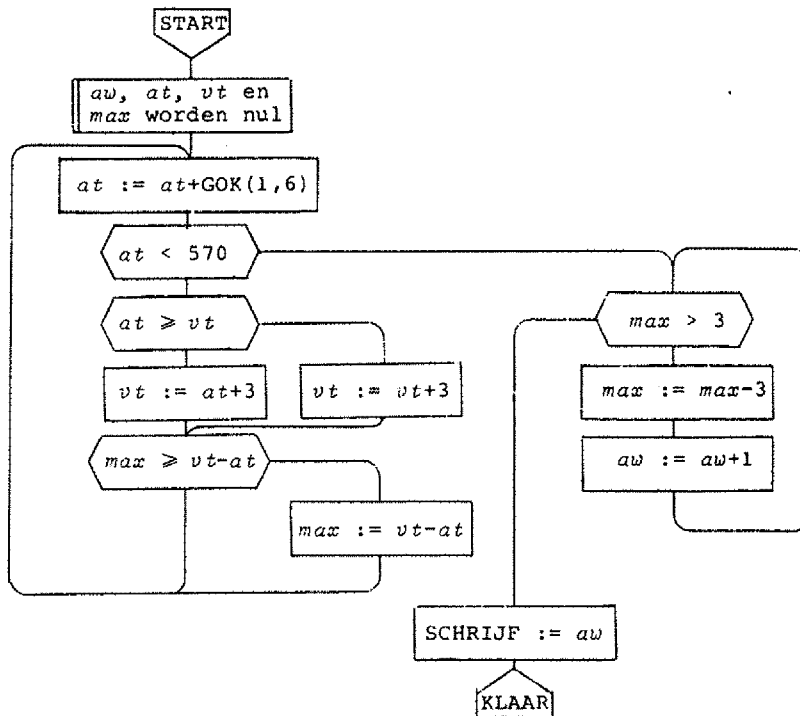
at = aankomsttijd van een auto

vt = vertrektijd van een auto, in het schema vanaf vt :=

Voordien is het de vertrektijd van de voorgaande auto.

max = de langst voorkomende tijd tussen aankomst en vertrek van eenzelfde auto

aw = aantal wachtende auto's.



De homogene verdeling van de tijden tussen twee klanten door GOK(1,6) doet onnatuurlijk aan (zie ook opmerking aan het eind van Compk 18.3), want het is allerm minst zeker dat binnen elke periode van 6 minuten tenminste één klant arriveert. Beter is uit te gaan van de kans op binnenkomen per seconde en wel als volgt:

De kans op binnenkomen in de 1^e sec. = $\frac{1}{60} \times \frac{1}{6}$

De kans op geen klant in de 1^e sec. = $1 - \frac{1}{360}$

De kans op geen klant in de 1^e 2 sec. = $(1 - \frac{1}{360})^2$

De kans op geen klant in de eerste n seconden =

$(1 - \frac{1}{360})^n = ((1 - \frac{1}{360})^{360})^{\frac{1}{360}n} \approx (e^{-1})^{\frac{1}{360}n} = e^{-\frac{1}{360}n}$

De kans P op een tussentijd van n seconden neemt sterk af bij grotere waarden van n. Men kan dit simuleren met behulp van de inverse funktie van $y = e^{-\frac{1}{360}x}$ nl. $x = -360 \ln y$. Bij een regelmatig verdeelde y is de tussenperiode x volgens onze eisen verdeeld.

Kortom: $x := -360 \text{ LN}(\text{GOKC})$ geeft een betere gesimuleerde tussentijd (in seconden uitgedrukt).

2.2 TIJDSHEMA

Voor het behandelen van de leerboekjes Compk. 1 en 2 is het volgende tijdschema redelijk gebleken. Het is gebaseerd op één wekelijks lesuur. Hierbij bedenke men dat niet alle vraagstukken uit dit boekje behandeld hoeven te worden. Daarom volgt onder het tijdschema een lijst van vraagstukken, die als facultatief beschouwd kunnen worden en door de meeste leerlingen kunnen worden overgeslagen.

Compk. onderwerp	aantal lessen	paragr. nummer	tijdstip cursusjaar
Inleiding, algoritme, blokschema computer	2	1-3	begin sept.
Geheugen en rekenop- drachten	3	5	eind sept.
Schrijven en lezen	2	6	half okt.
programme- ren	3	7	eind okt.
De vertak- king	3	8	eind nov.
De cyclus	4	9	begin jan.
De rij	3	10	begin febr.
formules + giro	3	12,13	begin mrt.
personeels- adm.	3	14	apr.
keuze onderwerp	uit	15-20	restant

(vervolg)

Wrks hoofdstuk	aantal lessen	tijdstip cursusjaar
7	1½	
8	2	eind nov.
9	4	begin jan.
10	2	eind jan.
11	3	eind febr.
12	3	eind mrt.
13	2	apr.
herhaling en keuzenonderwerp restant		

Facultatieve opgaven

Compk.	3.12	3.13	8.10	8.11	8.12
	8.13	9.3	9.6	9.10	9.25
	9.26	10.3			

Tijdschema werkschrift

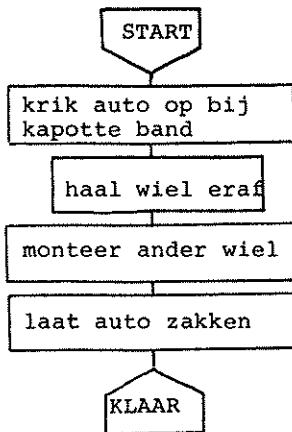
Wrks hoofdstuk	aantal lessen	tijdstip cursusjaar
1	2	begin sept.
2	1½	
3	1½	eind sept.
4	1	
5	2	
6	1½	eind okt.

2.3 OPGAVEN

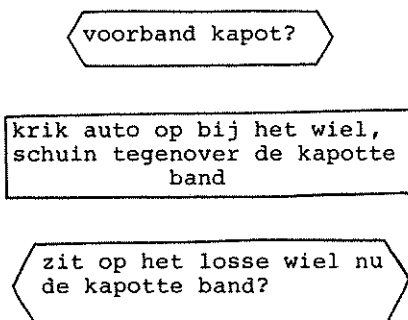
In de voorgaande jaren zijn door de experimenterende docenten opgaven uitgewisseld, die gebruikt werden bij proefwerken e.d. Deze vraagstukken hebben in de meeste gevallen de moeilijkheidsgraad van die uit Compk. Ze zijn hier globaal naar onderwerp gerangschikt en gewijzigd in die zin, dat getallenband gewijzigd is in getalkaarten. Dit laatste in verband met het gebruik van schrapkaarten. Er is afgezien van het vermelden van de herkomst van elk vraagstuk, beslist zonder hiermee ondankbaarheid te willen tonen aan alle trouwe inzenders.

Blokschema's

1) Een automobilist krijgt pech, hij moet een wiel verwisselen want hij heeft een lekke band. Dan gaat hij als volgt te werk:



Er is nu aan automobilist die het ingewikkelder wil doen. Hij wil graag vóór goede banden hebben, en zijn reserveband is niet meer zo best. Als de kapotte band dus vóór is, dan wil hij dat wiel verwisselen met een achterwiel en het reservewiel moet dan achter komen te zitten. Als de kapotte band achter is, dan wil hij die gewoon verwisselen met het reservewiel. Maak voor deze automobilist een blokschema. In dat blokschema mag je gebruik maken van de hokjes, die ook in het bovenstaande schema voorkomen, en verder mag je nog gebruik maken van de hokjes:



2) Een zeker iemand moet een aantal monsters frankeren. Op een monster van 100 g of minder moet 20 ct; op monsters die zwaarder zijn dan 100 g, maar lichter dan 200 g of juist 200 g, moet 50 ct; voor monsters die zwaarder zijn dan 200 g moet het porto eerst worden berekend. Schrijf een blokschema voor dit frankeren, waarin behalve het beginblok en het eindblok, alleen blokken met de volgende vragen en opdrachten voorkomen :

gewicht $\leq$ 100 g?
 gewicht $\leq$ 200 g?
 zijn er nog meer monsters?
 frankeer met 20 ct
 frankeer met 40 ct
 bereken porto
 frankeer met berekend porto
 neem een monster

3) Een apotheker neemt een hulpje om zijn flessenvoorraad te sorteren. Vuile flessen moet hij weggooien. In bak 1 moeten de flessen met $inh \leq 25$ cc. In bak 2 met $inh \leq 100$ cc, maar meer dan 25 cc. In bak 3 met $inh \leq 250$ cc, maar meer dan 100 cc. In bak 4 met inhoud groter dan 250 cc. Gebruik alleen de volgende opdrachten en vragen:

fles vuil?
 gooi weg
 pak fles
 lees inhoud op flesbodem
 $inh > 25$ cc
 $inh \leq 100$ cc
 $inh > 250$ cc
 zijn er nog flessen
 in bak 1
 in bak 2
 in bak 3
 in bak 4

4) Een zekere automobilist laat telkens na 3000 km rijden olie verversen (oude olie er uit en nieuwe er in). Als het langer duurt dan twee maanden voor hij 3000 km gereden heeft, dan laat hij al na twee maanden olie verversen. Elke dag controleert hij het oliepeil en als de stand te laag is, laat hij olie bijvullen. Als de olie ververst moet worden, laat hij op die dag geen olie bijvullen.

Maak een blokschema met de vragen en opdrachten:
meer dan 3000 km gereden?
oliepeil te laag?
laat olie verversen
meer dan twee maanden gereden?
laat olie bijvullen

5) Een zeker iemand laat een partij appels keuren op kleur, smaak en gewicht. Zijn smaak, kleur en gewicht goed, dan is de partij eerste soort. Zijn er twee van de drie goed, dan is de partij tweede soort. Alleen smaak goed, alleen kleur goed, alleen gewicht goed of niets goed, dan is de partij derde soort.

Schrijf een blokschema met de vragen en opdrachten:

bepaal gewicht, kleur en smaak
gewicht goed?
kleur goed?
smaak goed?
partij is eerste soort
partij is tweede soort
partij is derde soort

6) Een zekere leerling moet voor een proefwerk een aantal stellingen leren. Telkens als hij een stelling geleerd heeft, zegt hij die op. Gaat het goed, dan gaat hij de volgende stelling leren. Gaat het fout, dan leert hij dezelfde stelling nog eens. Schrijf een blokschema met de vragen en opdrachten:

leer de eerste stelling
leer deze stelling nog eens
zeg de stelling op
stelling goed opgezegd?
leer de volgende stelling
zijn er nog stellingen te leren?

7) Maak een blokschema voor een automaat, die koffie levert als je er een kwartje in doet en voor voor een dubbeltje een beker melk geeft. Gebruik de vragen en opdrachten:

zijn er nog bekertjes ?
is het muntstuk een kwartje?
zet een beker klaar
geef muntstuk terug
zet koffiekraan even open
is er nog melk?
is er nog koffie?
zet melkkraan even open

8) Iemand oefent het hoogspringen door de lat op 1 meter hoogte te stellen. Na elke goede sprong wordt dan de lat 5 centimeter hoger gezet. Een foutsprong mag altijd éénmaal over.
Schrijf een blokschema met de vragen en opdrachten:

zet lat op 1 meter hoogte
spring
sprong fout?
zet lat 5 cm hoger
oefening is klaar
eerste fout op deze hoogte?

9) Een zekere leraar geeft op het kerstrapport g (goed), v (voldoende), t (twijfelachtig) of o (onvoldoende). Hij heeft van elke leerling één *proefwerk*-cijfer, dat cijfer is x. Hij heeft verder van elke leerling een *indruk*, die goed of slecht is. Hij gaat nu als volgt te werk:

Er komt een g op het rapport als $x > 7$ (dus $x = 8$ of 9 of 10, ongeacht de indruk).

Er komt een v op het rapport als:

$x = 7$, ongeacht de indruk
 $x = 6$, en de indruk is goed

Er komt een t op het rapport als:

$x = 6$, en de indruk is slecht
 $x = 5$, ongeacht de indruk
 $x = 4$, en de indruk is goed.

Er komt een o op het rapport als:

$x = 4$, en de indruk is slecht
 $x < 4$ (dus $x = 1$ of 2 of 3, ongeacht de indruk)

Maak nu een blokschema voor die leraar. Volgens dat blokschema moet hij alle leerlingen hun rapportbeoordeling geven. In de hokjes mogen staan:

$x < \text{cijfer?}$	neem eerste leerling
$x = \text{cijfer?}$	neem volgende leerling
$x > \text{cijfer?}$	alle leerlingen gehad?
indruk goed?	g op rapport
START	v op rapport
KLAAR	t op rapport
	o op rapport

10) Een partij bloemkolen moet over vier kisten worden verdeeld. In kist 1 moeten bloemkolen lichter dan 1 kg. In kist 2 moeten bloemkolen lichter dan $1\frac{1}{2}$ kg, maar zwaarder dan 1 kg of precies 1 kg. In kist 3 moeten bloemkolen lichter dan 2 kg, maar zwaarder dan $1\frac{1}{2}$ kg, of precies $1\frac{1}{2}$ kg. In kist 4 moeten bloemkolen van 2 kg, of zwaarder
Maak een blokschema van het sorteerwerk en schrijf daarbij in de hokjes:

pak een bloemkool
 weeg de bloemkool
 gewicht ≥ 2 kg?
 gewicht < 1 kg?
 gewicht $< 1\frac{1}{2}$ kg?
 nog meer bloemkolen te sorteren?
 in kist 1
 in kist 2
 in kist 3
 in kist 4

$$y = \frac{3}{\frac{1}{a^3} + \frac{1}{b^3} + \frac{1}{c^3}}$$

Achtereenvolgens moeten a, b, c, x en y worden afgedrukt.

11) In emmer A zitten kaartjes; op elk kaartje staat een *natuurlijk* getal. Alle getallen zijn *verschillend* en *groter dan nul*. In emmer B zit niets. Iemand moet het *grootste* getal, dat zich in emmer A bevindt, op een bord schrijven. Maak een blokschema, door gebruik te maken van de volgende blokken:



Eenvoudige rekenschema's of programma's

12) Een klas heeft 6 leerlingen. Hun leeftijden staan op getalkaarten. Maak een blokschema dat die getalkaarten leest, en dat de som en het gemiddelde van alle leeftijden schrijft.

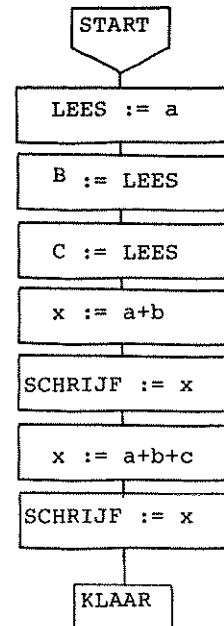
13) Schrijf een blokschema dat de getallen a, b en c van een getalkaart laat lezen en berekent:

$$x = \frac{a^2}{b^2} + \frac{b^2}{a^2} + \frac{c^2}{a^2}$$

14) Iemand heeft een programma geschreven en daarin onder andere de volgende instructies gebruikt. Welke ervan zijn goed en welke zullen niet door de computer worden aanvaard.

a := b : c	a := -15 × jantje
a := b/c	a := jantje × -15
start	17 := 18.5 - 1.5
cruyff := gevloerd	SCHRIJF := schrijf
klas 4 := kampioen	SCHRIJF := 16 + 84
a := 3,5 + 26	k := lees
KLAAR	LEES := x
x := y/z - a	b := schrijf

15) Iemand heeft een blokschema gemaakt voor het lezen van 3 getallen, te weten a, b en c. Hij wil 2 getallen schrijven nl. de uitkomsten van de optellingen a+b en a+b+c. Het blokschema ziet er zo uit:



Je ziet natuurlijk wel dat er fouten in dit schema zitten. Schrijf het blokschema en verbeter daarbij de fouten. Zet erbij hoeveel fouten erin zitten.

16) Maak een blokschema voor een computer. De computer moet twee getallen, a en b, lezen; dan x uitrekenen en tenslotte x schrijven; de formule voor x is:

$$x = \frac{a^2 + 2ab + b^2}{a^2 - 2ab - b^2}$$

Controleer het programma voor een getalkaart, waarop de getallen 3 en 5 staan.

Extra vraag: als a en b nu eens breuken zijn, bijv. a = 2.6387 en b = 0.0193, moet het blokschema daarvoor dan anders worden geschreven?

17) Als een onderstaande bewering een opdracht is door de computer wordt geaccepteerd, zet er dan 'ja' achter, anders 'nee'.

```
ab := a * b      15 := 4 * 5
a := 5 + a      das := das * das
15 := 3 * 5      jan := ziek
15 = 3 * 5
```

18) Schrijf een blokschema, dat (door jou zelf bedachte) getallen in de GP's a en b stopt en daarna berekent:

$$\frac{a^3 - b^3}{a+b}$$

Schrijf onder het blokschema in welke GP het antwoord zit.

19) Op een zekere school wordt het overgangsrapportcijfer berekend door op te tellen: het cijfer van het eerste rapport, tweemaal het cijfer van het tweede rapport en tweemaal het cijfer van het derde rapport en deze som te delen door 5. Schrijf een blokschema, dat (door jou zelf bedachte) cijfers van de eerste drie rapporten in GP's stopt en het overgangsrapportcijfer berekent. Schrijf onder het blokschema in welke GP dit overgangsrapport zit.

20) Als er bij het huiswerk geen repetitie is, dan zijn er 2 gevallen:

1. Er is huiswerk voor meer dan 4 vakken.
Dan wordt aan elk vak 40 min. gewerkt.
2. Er is huiswerk voor 4 of minder vakken.
Dan wordt aan elk vak 50 min. gewerkt.

Als er bij het huiswerk wel een repetitie is, dan wordt aan elk ander vak 40 min. gewerkt, en aan de repetitie 90 min.

Schrijf een blokschema en een programma, dat gegevens van de getalkaart leest en berekent, hoeveel minuten aan het huiswerk is gewerkt, en

afdrukt: 'Aan het huiswerk is ... minuten gewerkt'. In plaats van de puntjes moet dan het door de computer berekende getal worden afgedrukt.

Op de getalkaart staat:

1. Een gegeven dat uitdrukt of er wel of geen repetitie is. Kies hiervoor zelf iets en schrijf onder de oplossing wat je hebt gekozen.
2. Het aantal niet-repetitievakken.

Schrijf zinvolle getallen onder je programma.

21) Iemand (die een rijbewijs heeft) huurt een dag een auto. Voor 25 gulden mag men 100 km rijden. Rijdt men meer, dan moet men voor elke km. die men meer rijdt dan 100 km, 15 cent extra betalen. Is men jonger dan 23 jaar, dan moet men 5 gulden meer betalen.

Schrijf een blokschema en een programma, dat het aantal gereden km's en de leeftijd van de huurder van de getalkaarten leest en afdrukt:

'De kosten bedragen gulden'.

Schrijf zinvolle getallen onder aan het programma.

22) Voor ambtenaren geldt ongeveer de volgende pensioenregeling. De pensioengrondslag is het grootste van: (het salaris over het laatste dienstjaar) en (het gemiddelde van de salarissen over de laatste drie dienstjaren).

Voor elk dienstjaar krijgt men als pensioen $1\frac{1}{4}\%$ van de pensioengrondslag (dus: als iemand 10 dienstjaren heeft dan krijgt hij:

$$10 \times \frac{1\frac{1}{4}}{100} \text{ de pensioen-}$$

grondslag), maar nooit meer dan 70% van de pensioengrondslag.

Schrijf een blokschema en een programma, dat de salarissen over de laatste 3 dienstjaren en het aantal dienstjaren van de getalkaarten leest (het salaris over het laatste dienstjaar moet op de derde plaats staan) en het pensioen berekent en afdrukt. Schrijf weer zinvolle getallen onder aan het programma.

```
23)      a := 5      mag wel/niet
        kees := a      mag wel/niet
        JAN := p      mag wel/niet
        ab := 8      mag wel/niet
        c := 1,4      mag wel/niet
        p := a:b      mag wel/niet
        a*b := 6      mag wel/niet
        a := a*8      mag wel/niet.
```

24) welke waarden hebben de variabelen a en b na afwerking van onderstaand programma? Maak een geheugenschema.

```
START
a := 3
b := a+a
a := b/a
a := a*a
b := a-b
b := a/b
KLAAR
```

25) Schrijf een blokschema voor het berekenen van:

$$\frac{347^3 - 49^3}{347 - 49}$$

(niet zelf de uitkomst berekenen !)

```
26) START
TEKST := "....."
a := LEES
b := LEES
b := b*b
a := 5
a := a+b
NR
SCHRIJF := a
KLAAR
```

a. Wat wordt er afgedrukt door het programma hierboven als er op de getalkaarten staat:

+3+6 ?

b. En wat als er staat:

+13+6 ?

c. Wat wordt er afgedrukt als er +13+6 op de getalkaarten staat en we laten de derde regel uit het programma (a := LEES) weg?

27) Op een zekere getalkaart staan drie getallen, achtereenvolgens a, b en c. Schrijf een programma dat deze getallen leest en het volgende verricht:

indien $a \geq b+c$, dan het getal 1 schrijven
indien $a < b \times c$, dan het getal 2 schrijven
indien $b > c$, dan het getal 3 schrijven
indien $a^2 \geq b^2 - c^2$, dan het getal 4 schrijven.

Indien aan geen der 4 voorwaarden wordt voldaan, dan het getal 0 schrijven.

Voorbeeld:

als a = 15 b = 5 c = 12

wordt er voldaan aan de 2^e en de 4^e voorwaarden en schrijft de computer dus op papier:

2 4

28) Schrijf een programma dat twee getallen (a en b) van de band laat lezen en berekent:

$$x = \frac{a^2 + 2ab + b^2}{a^2 - 2ab + b^2}$$

en deze x laat afdrukken. Doe dit in zo weinig mogelijk opdrachten.

29) Op een zekere getalkaart staan twee repetitie cijfers. Schrijf een programma dat deze twee cijfers laat lezen en laat afdrukken:

goed (als het gemiddelde van de repetitiecijfers of meer is) of

onvoldoende (als dit gemiddelde minder dan 5 is) matig (voor het geval dat niet "goed" of "onvoldoende" wordt geschreven).

Cyclus

30) In deze les zullen we proberen het verwerken van een computerprogramma na te bootsen. We gaan in groepjes van zes bij elkaar zitten. In elk groepje wordt een leider/leidster gekozen, die het programma leest en de opdrachten doorgeeft aan de andere leden van de groep. Er is een rekenaar(sters) die alle berekeningen uitvoert en door de leider worden de resultaten doorgegeven aan de betrokken personen. Er is in een 'dubbelrol' een lezer en een schrijver, die de LEES-, resp. de SCHRIJF-opdrachten uitvoert. Verder zijn er in elke groep drie geheugenplaatsen, die de getallen onthouden (noteren) zoals die in het programma voorkomen.

Verwerk op deze manier het volgende programma:

```
START
TEKST := "imitatiecomputer"
NR
som := 0
teller := 1
3  getal := LEES
   som := som + getal
   teller := teller + 1
   ALS teller > 10 DAN 7 ANDERS 3
7  TEKST := "de som van de getallen is"
   SCHRIJF := som
   KLAAR
   + 1123 + 11 + 23.22 + 27.15 + 21 + 30 + 21
   + 38.4 + 1 + 200
```

31)

Op zekere getalkaarten staat een aantal positieve getallen afgesloten door een nul. Schrijf een programma dat deze band leest en de positieve getallen onder elkaar afdruckt met naast elk getal het omgekeerde van dat getal. (Men lette bij de correctie op het feit dat niet het omgekeerde van nul berekent wordt).

32) Als men 8 halveert krijgt men 4, halveert men het resultaat weer dan krijgen we 2. Na 7 maal halveren hebben we $\frac{1}{16}$.

Na voldoende aantal malen halveren wordt het resultaat kleiner dan $\frac{1}{1000}$.

Nu de opgave.

Schrijf een programma waarbij een onbekend positief getal van een getalkaart wordt gelezen en waarbij als antwoord verschijnt hoe vaak, te beginnen met dit getal, het tenminste gehalveerd moet worden om een resultaat kleiner dan $\frac{1}{1000}$ te krijgen.

(Men lette bij correctie op het feit dat het te lezen getal kleiner dan 0.001 kan zijn).

33) Een zekere winkelier maakt de volgende getallen op getalkaarten:

het 1e getal = de prijs van artikel 1 per stuk

het 2e getal = het aantal stuks van artikel 1, dat verkocht is

het 3e getal = de prijs van artikel 2 per stuk

het 4e getal = het aantal stuks van artikel 2, dat verkocht is

enz.

De rij getallen wordt afgesloten door -1.

Schrijf een programma dat de omzet (dat is de totale hoeveelheid geld die hij door de verkoop heeft ontvangen), van deze winkelier berekent.

34) Op zekere getalkaarten staat een rij getallen $\neq 0$, gevolgd door een nul. Schrijf een programma, dat de som van al de getallen groter dan tien berekent en afdruckt.

35) Op zekere getalkaarten staat een rij getallen $\neq 0$, gevolgd door een nul. Lees elk getal en druk af PLUS als dat getal > 0 is en MIN als dat getal < 0 is.

36) Op zekere getalkaarten staat een rij getallen $\neq 0$, gevolgd door een nul. Schrijf een programma dat onderzoekt of er een positief getal bij is. Dus je drukt af JA als er wel een getal > 0 bij is en NEE als er geen getal > 0 bij is.

37) Weer dezelfde getalkaarten. Lees elk getal in, en druk af GROTER als dat getal > 1 en TUSSEN als dat getal a zo is dat $-1 \leq a \leq 1$ en KLEINER als dat getal < -1 is.

38) Op zekere getalkaarten staat een even aantal getallen gevolgd door een nul. Schrijf een programma, dat achtereenvolgens afdruckt:

de grootste van het 1e en 2e getal

de grootste van het 3e en 4e getal

de grootste van het 5e en 6e getal enz.

39) Een steen valt van een 100 meter hoge toren. We willen de computer laten berekenen na hoeveel tijd de steen op de grond is aangekomen. Dit gaat als volgt:

De afstand a van de steen tot de grond is $100 - 5t^2$; hierin is t de tijd die de steen valt.

Laat in een programma t starten bij 0 seconden, laat t daarna steeds 0,1 sec. toenemen. Bereken bij elke t de afstand a . Laat het programma stoppen als a negatief wordt en laat de tijd t op dat moment afdrukken.

40) Op getalkaarten staat een aantal gehele getallen die groter zijn dan nul en kleiner dan 1. Deze rij getallen wordt op de band afgesloten met het getal nul. Schrijf een programma voor het volgende.

Lees de getallen en tel hoeveel getallen er liggen in de intervallen 1 t/m 25, 26 t/m 50, 51 t/m 75 en 76 t/m 100. Wanneer er een 0 wordt gelezen moet de telling stoppen en de vier antwoorden (nl. de vier getelde hoeveelheden), worden geschreven.

RIJ

41) Op een band staan 20 getallen (gehele getallen van hooguit zes cijfers). Schrijf een programma dat deze band laat lezen en de getallen op de volgende wijze laat afdrukken:

1e getal	6e getal	11e getal	16e getal
2e getal	7e getal	12e getal	17e getal
3e getal	8e getal	13e getal	18e getal
4e getal	9e getal	14e getal	19e getal
5e getal	10e getal	15e getal	20e getal

42) Op zekere getalkaarten staan 1000 rapportcijfers (gehele getallen 1 tot en met 10). Schrijf een programma dat telt hoe vaak het cijfer 1 voorkomt, ook hoe vaak het cijfer 2 voorkomt, enz. tot en met de tien. Laat deze tien antwoorden afdrukken.

43) Op getalkaarten staan 100 getallen. Het zijn gehele getallen van hoogstens zes cijfers. Deze getallen moeten worden gelezen en in een afwijkende volgorde worden geschreven, namelijk tien getallen op één regel, in omgekeerde volgorde. Dus zo:

10e getal 9e getal 8e getal 1e getal
 20e getal 19e getal 11e getal

 1000e getal 999e getal 991e getal

44) Op getalkaarten staan de leeftijden van de leerlingen van een zekere school, die variëren van 11 tot 21 jaar. De rij getallen wordt afgesloten door een nul. Schrijf een programma, waarbij de kaarten éénmaal gelezen worden en waarin geteld wordt hoeveel leerlingen van 11 jaar er zijn, hoeveel leerlingen van 12 jaar er zijn, hoeveel van 13 jaar enz. tot en met 21 jaar.

45) Iemand wil (we weten niet waarom) de volgende getallen in een rij geheugenplaatsen hebben.

a(1)	a(2)	a(3)	a(4)	a(5)	a(6)	a(7)	a(8)	a(9)
1	2	3	4	5	4	3	2	1

Schrijf hiervoor een zo kort mogelijk programma.

46) Op zekere getalkaarten staan 20 getallen $a_1, a_2, \dots, a_{10}, b_1, b_2, \dots, b_{10}$. Schrijf een programma dat de waarde van $a_1 \times b_{10} + a_2 \times b_9 + \dots + a_{10} \times b_1$ berekent en afdruckt.

47) Op zekere getalkaarten staan 100 getallen $g_1, g_2, \dots, g_{100}$. Schrijf een programma voor het berekenen en afdrukken van $g_1 + g_{51}$, en van $g_2 + g_{52}$, en $g_3 + g_{53}$, ... $g_{50} + g_{100}$.

Opdrachten G. SCHRIJF(n,m). GOK

48) Op een zekere getalkaart staat een positief geheel getal. Schrijf een programma, zodat de computer afdruckt

JA indien het getal op de kaart wel door 3 of 4
 NEE indien het getal op de kaart noch door 3 noch door 4 deelbaar is.

49) Op zekere getalkaarten staat een rij positieve gehele getallen, afgesloten door een nul. Schrijf een programma, zodat de computer afdruckt

NEE indien geen enkel getal door 3 deelbaar is
 JA indien er wel een getal onder is, dat deelbaar is door 3.

50)

Geef een blokschema van het onderstaand programma. Geef met je eigen woorden weer, wat 't programma doet.

```

START
TEKST := "....."
NR
1  a := LEES
2  ALS a = 0 DAN 11 ANDERS 4
3
4  a := a/5
  b := G(a)
  ALS a = b DAN 1 ANDERS 8
8  NR
  TEKST := "NEE"
  NAAR 13
11 NR
12 TEKST := "JA"
13 KLAAR

```

Als de getalkaart er als volgt uitziet:

+ 15 + 20 + 25 + 0. Geef dan weer, wat er afgedrukt wordt. Als er op regel 3 NR wordt ingevuld, wordt er wel/niet iets anders afgedrukt. Als in regel 2 de 11 door een 12 vervangen wordt, wordt er wel/niet iets anders afgedrukt.

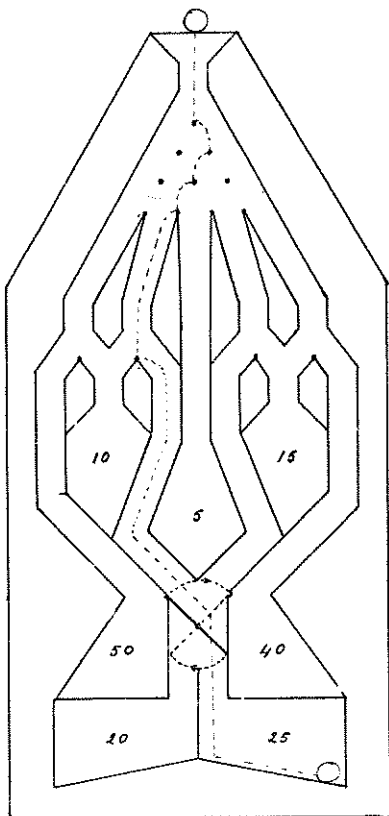
51) Op een zekere getalkaart staat een prijs in guldens, dus bijvoorbeeld 8.35 voor f 8,35. Schrijf een programma dat de nieuwe prijs laat berekenen bij een prijsverhoging van 8%. Alleen wanneer deze nieuwe prijs niet op gehele centen uitkomt, moet naar boven op hele centen worden afgerond. De nieuwe prijs moet geschreven worden in guldens.

52) Op zekere getalkaarten staat een onbekend aantal positieve getallen, afgesloten door een nul. Schrijf een programma dat de band laat lezen en deze getallen onder elkaar laat afdrukken, met naast elk getal het omgekeerde van dat getal, beide in drie decimalen nauwkeurig.

53) Op zekere getalkaart staat één getal, namelijk de rente die een bank geeft (bijvoorbeeld 5 voor 5% of 4.5 voor 4½%). Een bedrag van 100 gulden wordt op deze bank gezet. Na één jaar wordt de rente, afgerond op hele centen, erbij geteld. Het jaar daarop wordt van dit nieuwe bedrag de rente berekend en erbij geteld. Evenzo de volgende jaren. Laat de computer de rente van de kaart lezen en een lijst maken, waarin staat het te goed (het bedrag dat op de bank staat) na één jaar, na twee jaar, na drie jaar, enz. De lijst moet stoppen als het te goed boven 200 gulden gekomen is.

54) Uit een vat met een inhoud van x liter worden potten met een inhoud van y liter getapt. Deze potten worden verpakt in kisten van n potten. Schrijf een programma dat van een getalkaart de 3 getallen x , y en n leest en vervolgens afdrukt:
a) het aantal kisten (ak) dat verpakt kan worden;
b) het aantal potten (ap) dat overblijft;
c) de hoeveelheid (hv) die in het vat achterblijft, nadat de laatste volle pot eruit getapt is.

55) Een aantal balletjes wordt boven in het apparaat (zie tekening) één voor één losgelaten. Elk balletje vindt zijn weg naar één van de genummerde hokjes via pennetjes, kanaaltjes en soms via een klep. De balletjes vallen boven op de pennetjes en hebben daarna 50% kans om naar rechts en 50% kans om naar links te gaan. De klep verandert van positie, wanneer er een balletje is gepasseerd. In de vakken 20 en 25 komen dus of evenveel balletjes, of het verschil is 1. Schrijf een programma voor de simulatie van dit toestel, wanneer 100 balletjes gegooid worden. Laat de vulling van elk van de zeven vakjes afdrukken en bovendien de totaalscore, wanneer elk balletje de waarde krijgt die op het vakje staat aangegeven.



56) Schrijf een programma dat een aantal toevalsgetallen produceert. Het begingetal van 5 cijfers wordt van een kaart gelezen. Het toevalsgetal krijgen men door dit getal met 101 te vermenigvuldigen, bij het product 33333 op te tellen en dan de laatste vijf cijfers van het antwoord te nemen. Uit dit getal wordt het volgende toevalsgetal verkregen.

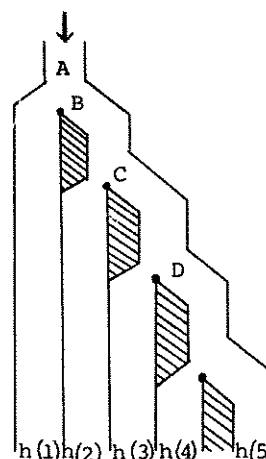
57) Schrijf een programma dat het getal n van een getalkaart leest en vervolgens n keer met een dobbelsteen gooit en afdrukt hoe vaak er 1,2,3 enz. is gegooid.

58) Schrijf een programma dat via toevalsgetallen een kolom van 13 wedstrijden invult van de voetbaltoto.

59) Schrijf een programma dat een frequentieverdeling maakt van de som van het aantal ogen bij worpen met twee dobbelstenen. Vergelijk het resultaat met een theoretisch berekende frequentieverdeling.

60) Het is mogelijk de computer een melodie te laten componeren. We gebruiken de acht tonen c, d, e, f, g, a, b, c^1 . Kies de eerste toon met behulp van de opdracht $GOK(,)$. Laat ook de lengte daarvan gokken (2,1 of $\frac{1}{2}$ tel). Schrijf een programma voor een melodie van 16 tellen. Laat het resultaat afdrukken in de vorm van $a2, c1, c^11, e\frac{1}{2}, \dots$

61)



Bij A vallen 100 knikkers binnen (zie figuur). Bij elk van de splitsingen B, C, D enz. heeft een knikker evenveel kans op naar links of naar rechts vallen. Schrijf een programma voor de simulatie van dit toestel en laat de vulling bepalen van de hokjes $h(1), h(2), \dots, h(5)$. Neem in dit programma ook op het maken van een liggend histogram in de vorm van

$h(1)$	x x x x x x
$h(2)$	x x x x
$h(3)$	x x x x
$h(4)$	x x
$h(5)$	x x

Bestanden

62) In een winkel worden 100 verschillende artikelen verkocht. Deze artikelen zijn genummerd van 1 t/m 100. De winkel wordt bevoorraadt vanuit een aangebouwd magazijn. In dit magazijn bevindt zich van elk der 100 artikelen een aantal stuks in voorraad. De voorraadadministratie van dit magazijn wordt gevoerd met behulp van een computer. Telkens wanneer iemand van een bepaald artikel een aantal stuks uit het magazijn naar de winkel brengt, schrijft de magazijnbediende twee getallen aan op een getalkaart, namelijk eerst het nummer van het artikel en daarna het aantal stuks dat hij heeft afgegeven. Aan het eind van elke dag schrijft hij aan het eind van de getallenrij het getal -1 aan.

Opmerkingen:

1. Van een bepaald artikel kan men meer dan eens per dag een aantal stuks uit het magazijn halen.
2. Het is niet zeker dat per dag van elk artikel een aantal stuks uit het magazijn wordt gehaald.

In het achtergrondgeheugen van de computer zijn onder andere de volgende bestanden aanwezig:
BESTAND(1) bestaat uit 100 getallen die aangeven: het aantal stuks dat van elk artikel in voorraad is (dus in het magazijn). Dus: het vijfde getal van BESTAND(1) is het aantal stuks dat van artikel nummer 5 in voorraad is.

BESTAND(2) bestaat uit 100 getallen die aangeven: de inkoopprijs van elk artikel. Dus: het vijfde getal van BESTAND(2) is de prijs van artikel nummer 5.

Opgaven hierover:

- a) Op zekere dag worden de inkoopprijzen van alle artikelen met 2% verhoogd. Schrijf een programma dat ervoor zorgt, dat de nieuwe prijzen van de artikelen in BESTAND(2) komen.
- b) Schrijf een programma dat aan het eind van een dag het voorraadbestand bijwerkt met behulp van de gegevens van de getalkaarten (dit programma moet er dus voor zorgen dat het oude BESTAND(1), zoals dat aan het begin van de dag was, wordt vervangen door een nieuw BESTAND(1) waarin voor elk artikel het aantal stuks staat dat aan het eind van de dag in het magazijn is).
- c) Ook moet het programma de nummers laten afdrucken van alle artikelen, waarvan aan het eind van de dag minder dan 30 stuks in het magazijn aanwezig zijn (dan worden deze artikelen die

avond bij de grossiers besteld en de volgende dag vóór openingstijd aan het magazijn bezorgd).

63) In BESTAND(1) staan 1000 saldi (geldbedragen) van rekeninghouders in de volgorde van de gironummers. Men wil onder elkaar de gironummers 1 t/m 1000 afgedrukt hebben, met naast elk nummer het saldo. Schrijf hiervoor een programma.

64) In BESTAND(3) staan de leeftijden van 1000 leerlingen van onze school. Schrijf een programma dat de gemiddelde leeftijd uitrekent.

65) In BESTAND(1) staan 1000 saldi (geldbedragen) van rekeninghouders in de volgorde van de gironummers (1 t/m 1000). Men wil onder elkaar afgedrukt hebben de saldi die groter zijn dan 30 gulden, met naast elk saldo het bijbehorende gironummer. Schrijf hiervoor het programma.

66) In de bestanden 101 t/m 107 van het politiearchief staan de 200 bekendste Nederlandse misdadigers als volgt geregistreerd:

Bestand 101 het geslacht (0=man, 1=vrouw)

Bestand 102 geboortejahr in vier cijfers

Bestand 103 lengte in centimeters

Bestand 104 in hechtenis op dit moment? (0=nee, 1=ja)

Bestand 105 aantal veroordelingen

Bestand 106 aard van het laatste misdrijf

(1=inbraak, 2=geweldpleging, 3=vermoordensdelict, 4=overige misdaden)

Er is op elk politiebureau een genummerde lijst met de 200 namen, in dezelfde volgorde als in de bestanden is aangehouden.

a) Er is vannacht een inbraak gepleegd door een (volgens getuigeverklaringen) man van minstens 1 meter 70 lang van in de dertig jaar (d.w.z. 30 ≤ leeftijd < 40).

Schrijf een programma voor het afdrucken van de nummers van de inbrekers (= mensen die de laatste keer voor inbraak veroordeeld zijn) die het misdrijf gepleegd kunnen hebben, rekening houdend met de getuigeverklaringen.

b) Uit de gegevens van het politiearchief wil men een statistiekje samenstellen en wel hoeveel er éénmaal zijn veroordeeld, hoeveel er tweemaal zijn veroordeeld, hoeveel driemaal, enz. Geen enkele

geregistreerde heeft meer dan 10 veroordelingen achter de rug. Schrijf een programma dat deze tien getallen laat schrijven.

c) Bedenk een manier om in één bestand van 200 getallen van alle misdadigers te registreren de aard van het misdrijf bij elke veroordeling (dus niet alleen de laatste). Je mag aannemen dat niemand meer dan 10 veroordelingen achter de rug heeft. Geef hierbij expliciet aan hoe iemand met vier veroordelingen wegens misdrijven +1,+1,+4,+3 in dit bestand geregistreerd staat.

Geef in een blokschema aan hoe deze registratie ontstaat als deze vier getallen van een getalkaart worden gelezen. Is de door jou bedachte manier geschikt om alle nummers te vinden van de personen, die wel eens vanwege misdaad + 2 veroordeeld zijn? Zo ja, hoe moet dat worden geprogrammeerd?

67) De bestanden 201 t/m 206 bevatten het bevolkingsregister van een klein dorp. Alle 1000 bewoners van de laatste tweehonderd jaar staan hierin vermeld. We nemen voor het gemak aan dat er geen bewoners van buitenaf in die tijd aan het register zijn toegevoegd. De registratie is als volgt:

Bestand 201 geslacht (0=man, 1=vrouw)

Bestand 202 burgerlijke staat (0=ongetrouwd, 1=getrouwd)

Bestand 203 geboortear (4 cijfers)

Bestand 204 in leven (0=nee, 1=ja)

Bestand 205 het nummer van de vader

Bestand 206 nummer van de echtgenoot (echtgenote)

(0 bij ongetrouwd zijn)

Er is een lijst waarop de volledige namen en adressen van de inwoners, genummerd in dezelfde volgorde als in de bestanden is aangeboden.

a) Aan het loket verschijnt een inwoner die een uittreksel uit het geboorteregister wil hebben, waarop zijn naam en de namen van zijn beide ouders staan vermeld. Het nummer van de inwoner wordt op een schrapkaart gezet. Schrijf een programma, waarin deze getalkaart wordt gelezen en dit nummer en de nummers van de beide ouders worden afgedrukt.

b) Schrijf een programma waarin een nummer van een getalkaart wordt gelezen, waarna het nummer van de grootmoeder aan moederszijde wordt gezocht en afgedrukt.

c) Schrijf een programma waarin wordt bepaald en geschreven hoeveel kinderen meneer nummer 65 heeft (gehad).

d) Voor een onderzoek (over de woonomstandigheden van vrijgezellen) heeft men de nummers nodig van de te enqueteren ongehuwde inwoners tussen de 30 en 50 jaar. Laat een lijst maken van de nummers met naast elk nummer de leeftijd.

e) Voor een statistiek wil men weten hoeveel inwoners er zijn van 0 tot 9 jaar, hoeveel van 10 tot 19 jaar, hoeveel van 20 tot 29 jaar, enz. enz., tot en met 99 jaar en hoeveel van 100 of ouder. Laat een lijst afdrucken van deze aantallen.

68) Vliegtuigreservering.

In bestand 41 staan de vrije plaatsen (=0) en gereserveerde plaatsen (=1) in de opstelling van Compk figuur 15.1. Schrijf een programma voor het reserveren voor 4 plaatsen naast elkaar. Indien dit niet mogelijk is, onderzoek dan of er wel vier willekeurige vrije plaatsen zijn. Als resultaat moet verschijnen :

"gereserveerd op plaatsen nummers:" of
"er zijn vier vrije plaatsen, maar niet naast
elkaar" of
"er zijn geen vier vrije plaatsen".

Denk om het bijwerken van het bestand.

69) Iemand beweert dat men een kans $\frac{1}{10}$ heeft op een tweeletterig Nederlands woord, als men twee willekeurig gekozen letters die niet medeklinkers zijn achter elkaar zet.

a) Schrijf een programma dat 100 tweelettercombinaties 'gokt' en afdrukt, zonder dat hierin combinaties van twee medeklinkers voorkomen. Door te tellen hoeveel Nederlandse woorden hierin voorkomen, kan bepaald worden of de stelling aannemelijk is of niet.

b) Neem aan dat de 100 tweelettercombinaties uit a) op schrapkaarten zijn gezet en dat bestand 103 de 65 goed Nederlandse tweeletterwoorden bevat in alfabetische volgorde, achter elkaar en zonder spaties. Schrijf een programma dat uitzoekt en afdrukt hoeveel Nederlandse woorden er onder de 100 gegeven tweelettercombinaties zitten.

c) Vraagstukken a) en b) zijn ook te combineren tot één programma dat alleen het laatst gevraagde aantal afdrukt.

2.4 ENKELE PROGRAMMA'S

Behalve extra opgaven, opgesomd in de vorige paragraaf, werden door experimenterende docenten ook wel leerlingenprogramma's ingestuurd, die opvielen door hun oplossing. Hieruit is een vrij willekeurige keus gemaakt, die U hieronder kunt aantreffen.

2.4.1 B_r_o_w_n_s_e_b_e_w_e_g_i_n_g

Een programma voor de simulatie van een Brownse beweging.

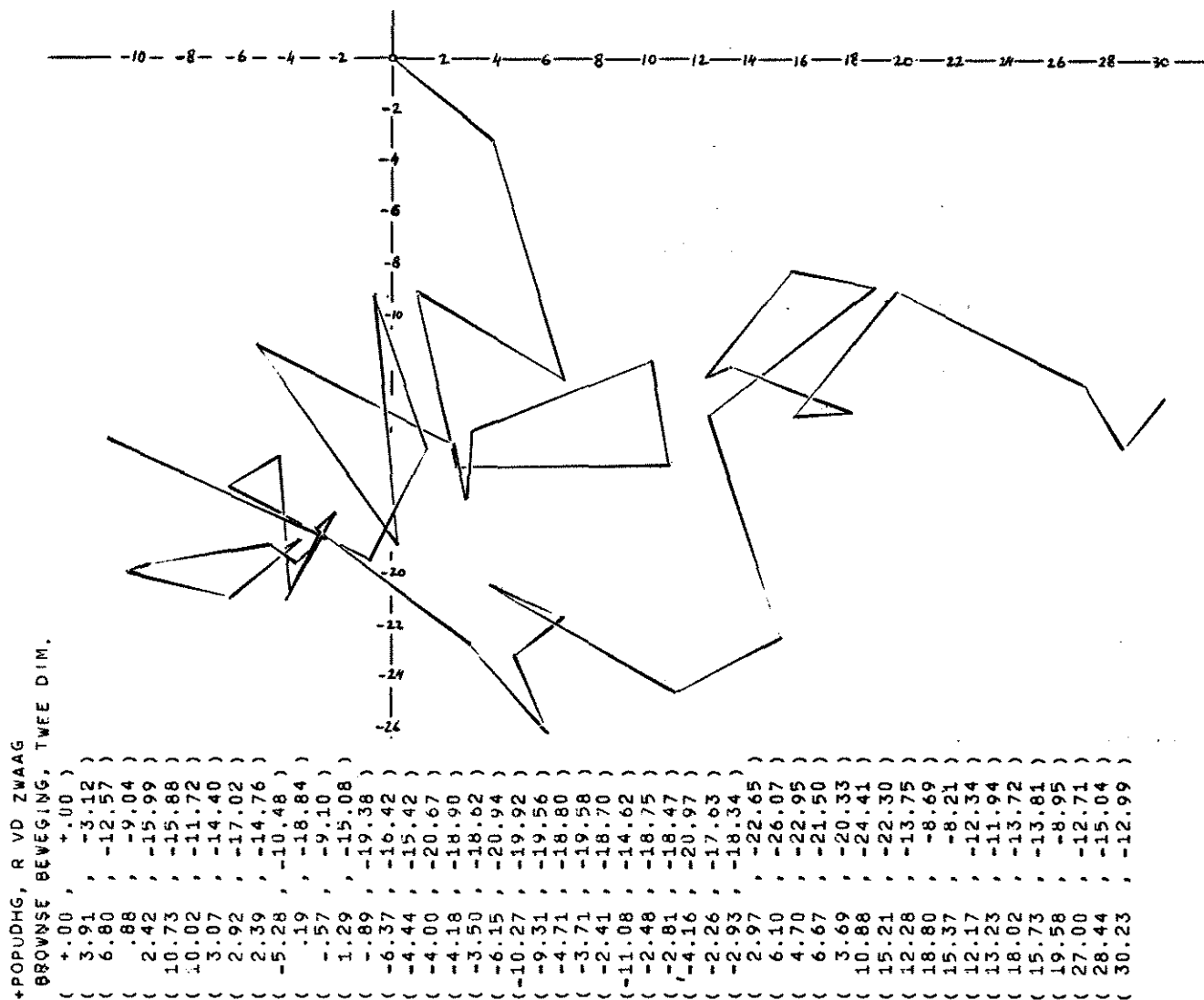
Het 'deeltje' beweegt zich rechtlijnig over een afstand r , waarbij r een toevalsgetal is met $0.00 \leq r \leq 10.00$. De richting fi (radialen) na een botsingsresultaat is een toevalsgetal $0.00 \leq fi \leq 6.3842$.

Onder het programma ziet U de posities in coördinaten en daarvan een met de hand getekende weergave.

```

START
TEKST:="+POPUDHG, R VD ZWAAG"
NR
TEKST:=" BROWNSE BEVEGING, TWEE DIM."
NR
T:=0
X:=0
Y:=0
1   T:=T+1
   SCHRIJFSYM:=78
   SCHRIJF(2,2):=X
   SCHRIJFSYM:=71
   SCHRIJF(3,2):=Y
   SCHRIJFSYM:=79
   ALS T=50 DAN3 ANDERS2
2   R:=GOK(0,1000)/100
   FI:=GOK(0,62831)/10000
   X:=X+R*CO$(FI)
   Y:=Y+R*SIN(FI)
   NR
   NAAR1
3   KLAAR

```



REKENT.IJD = 1.0 SEKONDE(N)

2.4.2. Iteratieve nulpuntbepaling

De functie $f: x \rightarrow x^2 - x - 1$ heeft een nulpunt tussen $x=1$ en $x=2$, want $f(1) < 0$ en $f(2) > 0$ en f is continu. Het volgende programma bepaalt het nulpunt, nauwkeurig tot het vierde cijfer achter de punt, met de zogenaamde halveringsmethode. Het nulpunt ligt steeds in het interval $[a,b]$. Bij iedere stap wordt de lengte van dit interval gehalveerd, door één van de grenzen van het interval te vervangen door $\frac{a+b}{2}$. De output bestaat uit een lijstje van deze intervalmiddens.

START	BENADERING VAN EEN NULPUNT VAN $x^2 - x - 1$
TEKST: "BEPALING VAN EEN NULPUNT VAN $x^2 - x - 1$ "	1.5000
OR	1.7500
A:=1	1.6250
B:=2	1.5625
1 X:=(A+B)/2	1.5937
SCHRIJF(X) NIET A	1.6094
OR	1.6172
ALS B-A > 0.0001 DAN 2 *MIDTUS 3	1.6211
1 X:=(X+B)/2	1.6191
2 ALS B-X > 0.0001 DAN 3 *MIDTUS 4	1.6182
3 A:=X	1.6177
2 *MIDTUS 1	1.6179
4 B:=X	1.6186
3 *MIDTUS 1	1.6180
5 KLAAR	1.6180

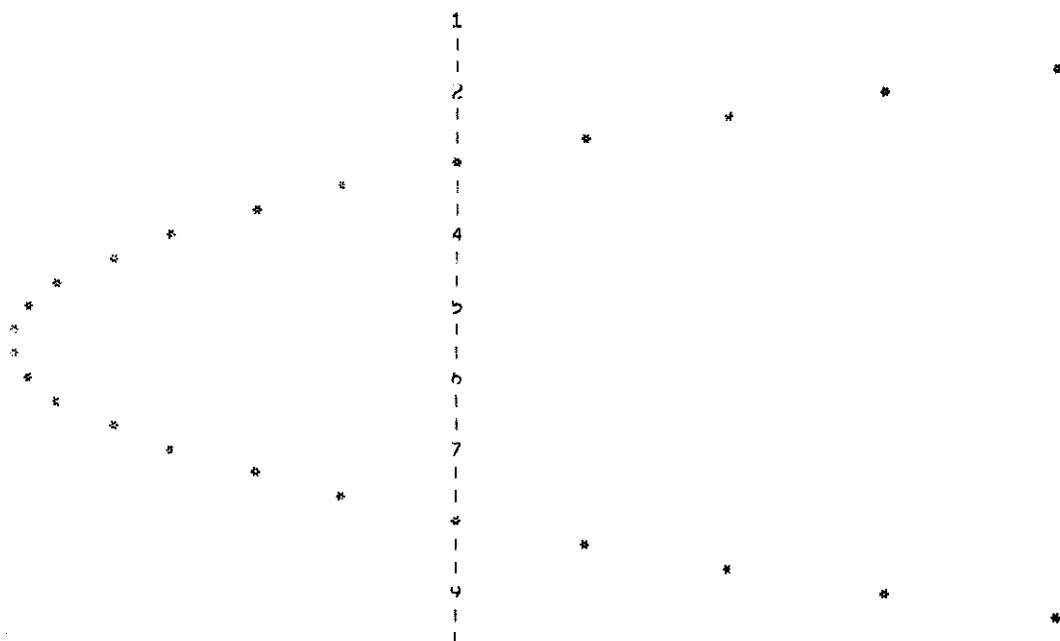
REKENTijd = 41 SEKONDE (V)

2.4.3. Een grafiek op de regeldrukker

Onderstaand programma geeft een "grafiek" van $f: x \rightarrow x^2 - 11x + 24$, met vertikaal staande X-as. Neemt men één positie op een regel en de regelafstand als horizontale respectievelijk verticale eenheid, dan ontstaat een vertekend beeld. Hier is voor de horizontale eenheid vijf posities en voor de verticale drie regels genomen, wat een redelijk "vierkant" beeld geeft.

Het aangeven van de roosterpunten op de as gaat hier wat onhandig (en onjuist voor $x > 9$).

GRAFIEK VAN $x^2 - 11x + 24$



```

START
TITEL:="GRAFIEK VAN X2=11X+24"
NR
N:=2
2 N:=N+1
  NR
  X:=N/5
  F:=X*X-11*X+24
  F:=F*5
  ALS F<0 DAN 3 ANDERS 6
3 SPATIE(F+49)
  SCHRIJFESYU:=82
  SPATIE(-F-1)
  ALS X=G(X) DAN 4 ANDERS 5
4 SCHRIJFESYU:=X
  NR 2
5 SCHRIJFESYU:=13
  NR 2
6 ALS F=0 DAN 7 ANDERS 8
7 SPATIE(49)
  SCHRIJFESYU:=92
  NR 2
8 SPATIE(49)
  ALS X=G(X) DAN 9 ANDERS 10
9 SCHRIJFESYU:=X
  NR 11
10 SCHRIJFESYU:=17
11 SPATIE(F=1)
  SCHRIJFESYU:=82
  ALS X>0.5 DAN 12 ANDERS 2
12 KLAAR

```

2.4.4

Een programma voor de berekening van $\lim_{n \rightarrow \infty} (1 + \frac{1}{n})^n$

n neemt hierbij achtereenvolgens de waarden, 2,4,8,16,...,2^q aan.

Het programma stopt op het criterium, dat twee achtereenvolgende antwoorden gelijk zijn tot 8 cijfers achter de komma.

```

START
oudea := 0
n := 1
2 n := 2 * n
  a := 1 + 1/n
  m := 1
3 a := a * a
  m := 2 * m
  ALS m = n DAN 4 ANDERS 3
4 SCHRIJF(8,0) := n
  SCHRIJF(2,8) := a
  NR
  ALS a-oudea < 0.00000001 DAN 6 ANDERS 5
5 oudea := a
  NAAR 2
6 KLAAR

```

2	2.25000000
4	2.44140625
8	2.56578451
16	2.63792850
32	2.67699013
64	2.69734495
128	2.70773902
256	2.71249162
512	2.71563200
1024	2.71695573
2048	2.71761848
4096	2.71795008
8192	2.71811594
16384	2.71819889
32768	2.71824035
65536	2.71826109
131072	2.71827146
262144	2.71827664
524288	2.71827923
1048576	2.71828182
2097152	2.71828182

3 Het 'ECOL' - rapport

In het nu volgende hoofdstuk vindt U een volledige beschrijving van de programmeertaal ECOL. Iets te volledig zelfs, want in enkele opzichten is men vooruit gelopen op de situatie. Met name het FUN-mechanisme en de MET-opdracht zijn op het moment van het verschijnen van dit boek nog niet in werkelijkheid toe te passen.

Dit hoofdstuk is ook onafhankelijk van dit docentenboek, als rapport, verschenen. Daarom wijkt de paragraafnummering af van die van overige hoofdstukken. Zelfs is dit hoofdstuk onderverdeeld in zes hoofdstukken, aangegeven met Romeinse cijfers. Hieronder vindt U een inhoudsopgave. Aan het eind van dit hoofdstuk vindt U ook een index om het zoeken te vergemakkelijken.

- INHOUD -

I	Taaldefinitie, vertaling, executie	15	Lege opdracht
1	Taaldefinitie	16	Invoer
2	Vertaling	16.1	LEES, LEESYM
3	Executie	16.2	BESTAND (x,x,x)
II	Globale vorm van ECOL programma's	16.3	BESTAND (x)
4	Structuur van een ECOL programma	17	Uitvoer
5	Labels, kommentaar, spaties	17.1	SCHRIJF (x,x)
5.1	Labels	17.2	SCHRIJF
5.2	Kommentaar	17.3	SCHRIJM
5.3	Spaties	17.4	SCHRIJFSYM
III	Context-regels en compiler actie	17.5	TEKST
6	Geheugenplaatsen, waarde, adres, descriptor	17.6	SPATIE(x)
7	Namen	17.7	SPATIE
8	Labels en sprongopdrachten	17.8	NR(x)
IV	Executie	17.9	NR
9	Inleidende opmerkingen	17.10	NP
10	Uitdrukkingen	17.11	BESTAND (x,x,x)
10.1	Getal	17.12	BESTAND (x)
10.2	Variabele	17.13	Opmerkingen uitvoer
10.3	Fun-roep	18	Standaardfuncties
10.4	Waarden van uitdrukkingen; prioriteiten	V	SUB en FUN
10.5	Volgorde bepaling primitiva	19	Inleiding
10.6	Waarde van aMb	20	SUB-beschrijving
11	Toekenning	21	SUB-roep
12	Sprongopdrachten	22	Informeel inleiding FUN mechanisme
13	RIJ-reservering	23	FUN-beschrijving
14	Cyclus	24	FUN-roep
		25	Recursiviteit
		VI	De ECOL syntax

DEFINIEREND RAPPORT VAN ECOL, EEN PROGRAMMEERTAAL VOOR HET VOORTGEZET ONDERWIJS

ECOL, samentrekking van Educational COmputer Language, is de programmeertaal die o.a. wordt gebruikt in de leergang "Computerkunde 1 en 2 voor a.v.o. en v.w.o." door C.A.Ch.Görts, S.G.van der Meulen, A.van der Sluis en J.R.Zweerus, uitgave Wolters-Noordhoff, Groningen 1970-1971.

Aan de opstelling van dit rapport werd, behalve door de vier auteurs van de genoemde leergang, meegewerkt door E.Lopez Cardozo, J.H.J.M.Truijens, G.A.Vonk.

I TAALEDEFINITIE, VERTALING, EXECUTIE

1. Taaldefinitie

ECOL is een taal om programma's voor een computer te formuleren. Bij het beschrijven van zo'n taal moet men enerzijds zeggen hoe correcte teksten in deze taal er uit zien, anderzijds wat voor computer-akties deze teksten beschrijven.

Hoe een correcte ECOL-tekst er uit ziet wordt voor een belangrijk deel vastgelegd in een syntax (zie VI). Omdat deze, ter wille van de eenvoud, weinig formeel is gehouden, is hij in feite wat te ruim, en zijn verdere inperkingen nodig. Deze worden in dit rapport gegeven in zgn. context-regels. Verwijzingen naar syntax-regels worden gegeven door [], verwijzingen naar context-regels door C...(C 8.1 bijv. staat in paragraaf 8).

In het rapport wordt voortdurend naar de syntax verwezen, en de lezer wordt dan ook uitgenodigd te beginnen met de inleiding tot de syntax te lezen.

2. Vertaling

Bij de computerverwerking van ECOL-teksten zijn twee fasen te onderscheiden, nl. de vertaalfase en de executiefase.

In de vertaalfase wordt de ECOL tekst door een voor ECOL gemaakt vertaalprogramma ("compiler") regel na regel, van boven naar beneden, en per regel van links naar rechts, gelezen en vertaald naar interne machine-opdrachten. We duiden deze leesvolgorde verder aan als tekstuele volgorde. Tijdens deze actie ziet de compiler voortdurend uit naar overtredingen van syntax en context-regels. Wanneer hij zo'n fout ontdekt wordt deze signaleerd, zoveel mogelijk met regelnummer en aanduiding wat er fout is. Na het ontdekken van een fout gaat de compiler rustig verder met lezen om te trachten nog meer fouten te ontdekken, zodat de auteur van het programma in één keer zoveel mogelijk fouten aan de weet komt. Evenwel kan een fout tot gevolg hebben dat foutmeldingen gegeven worden betreffende andere regels ECOL-tekst die in feite goed zijn.

Het is echter duidelijk dat als er eenmaal een fout is geconstateerd, niet tot executie van het programma kan worden overgegaan. Ook in de beschrijving van ECOL op de volgende pagina's duikt de compiler af en toe op. Strikt genomen is dit overbodig. Het kan echter bijdragen tot goed begrip van wat er gebeurt.

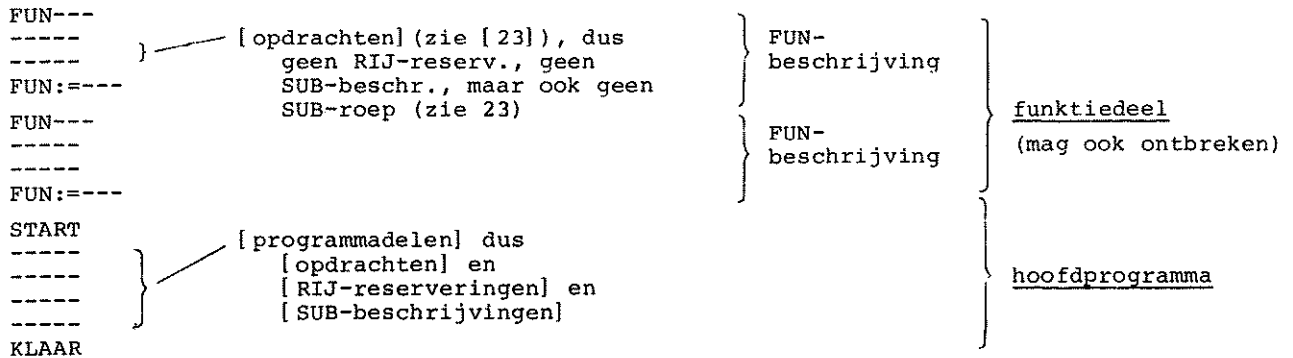
3. Executie

Als het programma bij vertaling "correct" bevonden is, wordt tot uitvoering (executie) overgegaan, d.w.z.: de computer gaat nu doen, wat de programmeur in zijn (taalkundig correct) programma voorschreef. Dit hoeft evenwel niet te betekenen, dat er nu ook gebeurt wat de programmeur beoogde. Er kan bijvoorbeeld iets uit een GP worden gehaald, voordat er iets in is gestopt; er kunnen RIJ-grenzen worden overschreden; er kan gevraagd worden een getal te lezen terwijl er geen getallen meer zijn, enz. Allemaal fouten die een vertaal-programma onmogelijk kan voorzien. We noemen dit executie-fouten; ook hiervan worden er vele signaleerd, weer met vermelding van regelnummer. Meestal heeft een executie-fout tot gevolg dat het programma wordt afgevoerd, d.w.z.: niet verder wordt geëxecuteerd.

II GLOBALE VORM VAN ECOL-PROGRAMMA'S

4. Structuur van een ECOL-programma

Uit de syntax (zie [20]) blijkt dat een ECOL-programma er als volgt uitziet.



Het funktiedeel, waarover in de leergang Computerkunde niet gesproken wordt, maakt het mogelijk, onafhankelijk van het hoofdprogramma een of meer rekenschema's (bestaande uit gewone ECOL-opdrachten "klaar te zetten", die dan door het hoofdprogramma kunnen worden gebruikt. Voor nadere verduidelijking zie 22 en volgende.

De indeling van een programmatekst in regels wordt in hoge mate door de syntax voorgeschreven. Overgang op een nieuwe regel programmatekst is in ECOL dan ook zeer betekenis-dragend. Zo mag men niet twee korte opdrachten op één regel zetten, of een lange formule op een volgende regel voortzetten.

5. Labels, kommentaar, spaties

5.1. Labels

Uit de syntax blijkt (zij het op een nogal ingewikkelde wijze), dat elke regel programma-tekst mag beginnen met hoogstens één [label], dat is een geheel getal zonder teken (zie [14]), eventueel gevolgd door dubbelpunt (zie [15]).

5.2. Kommentaar

Uit de syntax blijkt dat achteraan elke regel tekst een puntkomma, eventueel gevolgd door willekeurige symbolen mogen komen. Wat er achter de puntkomma staat is het zgn. kommentaar. De enige functie ervan is dat dit bij het afdrucken van de programmatekst mee wordt afgedrukt. Verder wordt er niets mee gedaan. Het dient om de tekst voor de menselijke lezer wat leesbaarder te maken.

In een programmatekst mogen op willekeurige plaatsen regels voorkomen die hetzij helemaal leeg zijn of alleen een label en/of kommentaar bevatten. Dit mag omdat [86] een "lege" FUN-beschrijving toelaat en [23] een "lege" opdracht. Wegens C 8.1 zijn daarom de labels op dergelijke regels tussen twee "echte" FUN-beschrijvingen geheel willekeurig, terwijl op dergelijke regels binnen een FUN-beschrijving of binnen het hoofdprogramma de labels moeten verschillen van de overige labels in de betreffende FUN-beschrijving resp. het hoofdprogramma.

5.3. Spaties

In ECOL-teksten mogen bijna overal spaties worden ingelast of weggelaten, zonder dat daardoor de betekenis, dat wil zeggen: de bij executie geproduceerde uitvoer, of de afgegeven foutmeldingen, verandert. De uitzonderingen staan in de inleiding tot de syntax:

- a) g e e n spaties mogen worden ingelast in hoofdletter-woorden (zoals START, LEES, ALS etc.);
- b) g e e n spaties mogen worden toegevoegd of weggelaten in een rijtje symbolen tussen aanhalings-teken, zoals achter "TEKST :=" kan voorkomen (dat is niet fout, maar daardoor verandert de geproduceerde uitvoer en dus de betekenis van het programma);
- c) tenminste 1 spatie m o e t er zijn wanneer hoofdletter-woorden op elkaar volgen.

Voorbeeld 5.1

```

7                ; demonstratietekst voor labels, kommentaar en spaties.
8  START
7:              ; deze nieuwe label 7 is toegestaan (zie 5.2 en C 8.1).
15 2            ; dit is label 152.
5 : x:  = LEES   ; lelijk "wordt-teken" maar wel goed.
    y := LEE S   ; fout, spaties in hoofdletterwoord.
9  ALS SIN(x) > 0.5 DAN 5 ; goed; spatie tussen ALS en SIN. In plaats van SIN(x) had er ook
                        ; mogen staan SIN ( x ).
5  TEKST := "jan hein" ; als men de spatie tussen jan en hein weglaat wordt er afgedrukt
                        ; janhein.
8  KLAAR        ; fout, label 8 was al uitgegeven in het hoofdprogramma; verder
                        ; mag op de regels volgend op KLAAR geen kommentaar meer komen.

```

III CONTEXT-REGELS EN COMPILER-AKTIE

6. Geheugen-plaatsen, waarde, adres en descriptor

Zoals in "Computerkunde I" staat, is een computer een apparaat dat getallen snel kan optellen, aftrekken, vermenigvuldigen, delen en vergelijken. We weten dat die getallen in het geheugen zijn opgeslagen; hoe dat technisch is gerealiseerd, doet hier niet ter zake. Wel van belang voor ons is:

Een plaats in het geheugen waar een getal kan worden opgeborgen, noemen we een Geheugen-Plaats (afgekort: GP).

Alle GP's in het geheugen zijn genummerd, het nummer van een GP heet zijn adres.

Het onderscheid tussen het getal in de GP (ook wel de inhoud of de waarde van de GP genaamd) en het adres van de GP speelt op verschillende plaatsen een rol in deze beschrijving van ECOL.

Aan elke [variabele] van de vorm [GP] (zie syntax [32]) in het hoofdprogramma wordt door de compiler een GP (dus een adres) toegekend.

Bij RIJ-en ligt de zaak ingewikkelder. Als men schrijft

RIJ(m:n)r of RIJSYM(m:n)r

dan krijgt men bij executie n-m+1 GP's ter beschikking:

r(m), r(m+1), -----r(n-1), r(n)

en elk van deze GP's heeft natuurlijk een adres. Maar de compiler weet niet welke adressen dit zijn omdat hij niet weet welke waarden m en n tijdens executie zullen hebben. Door sprongopdrachten kan het voorkomen dat een RIJ-reservering meerdere keren geëxecuteerd wordt met telkens andere onder- en bovengrens. Daarom zet de compiler voor elke rij een descriptor klaar, waarin tijdens executie geadministreerd kunnen worden

a) het begin-adres van de RIJ (dat is het adres van het eerste element)

b) de waarde van de onder-grens en de boven-grens,

c) gegevens die verder van belang kunnen zijn (zoals het al of niet gereserveerd zijn van de RIJ)

en hij vertaalt de RIJ-reservering in een aantal opdrachten die bij executie de nodige gegevens in de descriptor noteren. Een RIJ-reservering is dus wel degelijk een aktie.

Uit de syntax (zie [55]) blijkt nog dat onder- en bovengrens van een RIJ-reservering willekeurige [uitdr]'s (= uitdrukkingen, zie [34]) mogen zijn, zodat dus bijv. RIJ (p+1 : rxs)q toegestaan is.

7. Namen

Uit de syntax-regels [32], [33], [72] en [82] blijkt dat in een ECOL-programma verschillende soorten namen kunnen voorkomen:

[GP], [GP-rij], [SUB-naam] en [FUN-naam]

De schrijver van het programma weet natuurlijk wat er op elke plaats met een zekere [naam] wordt bedoeld, maar de lezer (in casu de compiler) weet dit niet. Vandaar de volgende context-regels

waaruit blijkt dat bij het in tekstuele volgorde lezen van een ECOL-tekst van elke [naam] die men ontmoet, direkt bekend is wat voor soort [naam] het is, ook als die [naam] nog niet eerder is voorgekomen. We volstaan hieronder met de context-regels voor het hoofdprogramma; de overeenkomstige regels voor het functie-deel vindt men in 23.

Context-regels voor het hoofdprogramma

C7.1 Een en dezelfde [naam] heeft overal in het hoofdprogramma dezelfde betekenis.

Gevolgen

- a) Een [naam] mag niet gebruikt worden in meer dan één der betekenissen [GP], [GP-rij], [SUB-naam], [FUN-naam].
- b) Als een [naam] een
[GP] is stelt hij overal dezelfde GP voor
[GP-rij] is stelt hij overal dezelfde RIJ voor
[SUB-naam] is verwijst hij overal naar dezelfde [SUB-beschrijving]
[FUN-naam] is verwijst hij overal naar dezelfde [FUN-beschrijving].
- c) Een [naam] mag niet meer dan eens op de plaats van [GP-rij], [SUB-naam] of [FUN-naam] in een [RIJ-reservering], [SUB-beschrijving] of [FUN-beschrijving] optreden.

C7.2 Een [naam] is een [FUN-naam] als hij eerder in de programmatekst op de plaats van [FUN-naam] in een [FUN-beschrijving] is opgetreden (zie [86]).

C7.3 Een [naam] is een [GP-rij] als hij eerder in de programmatekst op de plaats van [GP-rij] in een [RIJ-reservering] is opgetreden (zie [55]).

Gevolg

In de programmatekst mag een element van een zekere RIJ pas voorkomen nadat de reservering van deze RIJ verschenen is.

C7.4 Een [naam] is een [SUB-naam] als hij optreedt als [SUB-roep] of op de plaats van [SUB-naam] in een [SUB-beschrijving] (zie [70] en [71]). Een [SUB-naam] moet ook eenmaal op de plaats van [SUB-naam] in een [SUB-beschrijving] optreden.

Gevolg

In de programmatekst mag een [SUB-roep] voorkomen voordat de betreffende [SUB-beschrijving] is verschenen. N.B. Een [SUB-roep] is in de programmatekst gemakkelijk te herkennen: het is een [naam] die als zelfstandige [opdracht] optreedt (zie [70] en voorbeeld 7, regels 3 en 7).

C7.5 Een [naam] is een [GP] in alle overige gevallen.

Deze regels stellen bij een correct programma de compiler in staat alle optredende [namen] thuis te brengen. Wanneer tegen deze regels gezondigd wordt volgt een foutmelding, maar het kan nu verder voor de compiler wel moeilijk worden. Als een [naam] bijv. eenmaal verschenen is als [GP-rij] in een [RIJ-reservering] en vervolgens ook nog eens verschijnt zonder gevolgd te worden door een openingshaakje (dus in feite als [GP] of als [SUB-roep]), dan weet de compiler niet meer wat er nu eigenlijk bedoeld wordt (dit is een der oorzaken van het reeds in 2 gememoreerde verschijnsel dat een fout tot gevolg kan hebben dat "goede" stukjes tekst fout gemeld worden).

Voorbeeld 7.1

```

FUN f(x)    ; hiermee is "f" een [FUN-naam]
-----
-----
FUN :=w

START      ; hier begint het hoofdprogramma
a := LEES  ; hiermee is "a" een [GP]
RIJ(1:9)a  ; FOUT! "a" is al in gebruik genomen als [GP]
RIJ(1:a)r  ; in orde: "r" is hiermee een [GP-rij]
i :=a+3    ; hiermee is "i" een [GP]
y :=r(i)   ; hiermee is "y" een [GP], "r" wordt correct gebruikt als [GP-rij]
y :=u(i)   ; FOUT! "u" is nog niet verschenen in een [RIJ-reservering] en ook niet als
            ; [FUN-naam]
SUB klus   ; hiermee is "klus" een [SUB-naam]
-----
-----
END
y := klus  ; FOUT! "klus" wordt hier gebruikt als [GP], maar is al verschenen als
            ; [SUB-naam]
3 klus     ; in orde, een reeds beschreven SUB wordt geroepen
7 klos     ; GOED! "klos" verschijnt als een [SUB-roep], merk op dat het onmogelijk een
            ; [GP] kan zijn, "klos" is nu een [SUB-naam] en de compiler verwacht v ó ó r
            ; KLAAR de beschrijving van "klos"
y :=klos   ; dat is dus FOUT
SUB klos   ; daar is hij dan
-----
END
y :=f      ; FOUT! "f" is al verschenen als [FUN-naam] van een functie met een parameter
y :=f(a)   ; in orde, "f" wordt gebruikt in een [FUN-roep]
RIJ(1:a)f  ; FOUT! "f" is al verschenen als [FUN-naam] en staat niet meer ter beschikking
            ; als [GP-rij]
f          ; FOUT! "f" is al verschenen als [FUN-naam] van een functie met een parameter
KLAAR     ; dit programma komt nooit aan executie toe

```

8. Labels en sprongopdrachten

Zoals gezegd mag men elke programma-regel een label geven. Men is echter niet geheel vrij in de keuze van deze labels. Er geldt:

C8.1 Meerdere regels mogen hetzelfde getal als [label] hebben, mits maar alle labels in het hoofdprogramma verschillend zijn, evenals alle labels van een FUN-beschrijving (de labels in een SUB-beschrijving moeten dus ook verschillen van de labels in andere SUB-beschrijvingen en die in de rest van het hoofdprogramma).

Op deze regel is al geanticipeerd in 5.2 en voorbeeld 5.1.

Sprongopdrachten moeten verwijzen naar labels vooraan programmaregels. Ook hier is men weer niet geheel vrij. Er geldt:

C8.2 Een sprongopdracht die in het hoofdprogramma staat maar niet in een [SUB-beschrijving] mag alleen verwijzen naar een [label] in het hoofdprogramma maar niet in een [SUB-beschrijving].

C8.3 Een sprongopdracht die staat in een SUB- of FUN-beschrijving mag alleen verwijzen naar een label in dezelfde SUB- of FUN-beschrijving.

C8.4 Een sprongopdracht mag nooit verwijzen naar een [label] voor START, of voor een SUB- of FUN-beschrijving .

Als gevolg hiervan kan men een SUB of FUN alleen binnenkomen door de voordeur met een SUB- of FUN-roep, en alleen verlaten door de achterdeur met END resp. FUN:----

IV EXECUTIE

9. Inleidende opmerkingen

Wanneer de compiler een aangeboden ECOL-tekst correct heeft bevonden en heeft vertaald, kan het vertaalde programma in executie worden genomen, dat wil zeggen: worden uitgevoerd.

Bij executie van een ECOL-programma wordt begonnen met de [aktie] die tekstueel volgt op START. Na elke [aktie] wordt de tekstueel volgende [aktie] uitgevoerd, behalve bij sprongopdrachten, SUB- en FUN-roepen. Aangezien [SUB-beschrijvingen] geen [akties] zijn vallen zij buiten de executie-volgorde. Men mag dus rustig tussen twee [akties] een [SUB-beschrijving] zetten: die wordt bij executie overgeslagen. Men ziet dat de syntactische term [aktie] juist gekozen is: een [aktie] doet inderdaad iets, terwijl een [SUB-beschrijving] een rekenschema is dat door een [SUB-roep] geactiveerd moet worden.

In dit hoofdstuk wordt in meer detail besproken hoe een correct ECOL-programma wordt geëxecuteerd. We beginnen daarbij met te bespreken hoe waarden worden berekend uit formules. Daarna komen de verschillende akties ter sprake.

Afronden

In het volgende wordt nogal eens gesproken over afronding. Wanneer voor een willekeurig getal a en een geheel getal g geldt $g - \frac{1}{2} \leq a < g + \frac{1}{2}$ dan verstaan we onder het afronden van a het vervangen van a door g .

10. Uitdrukkingen

In [34] t/m [38] wordt recursief (voor de betekenis van dit woord zie inleiding syntax) vastgelegd wat een [uitdr] allemaal kan zijn. Zinnige samenstellingen van getallen, variabelen en FUN-roepen (de zgn. primitiva) met de operatoren $+$ $-$ $\times$ $/$ en M (machtsverheffing) en haakjesparen zijn toegestaan, mits geen twee van deze operatoren onmiddellijk op elkaar volgen. Dus niet $a \times -b$, ook niet $a \times -6$, maar wel $a \times (-b)$. Enige uitzondering: men mag i.p.v. M ook $\times \times$ schrijven.

Ingewikkeld voorbeeld:

$$r(i+j \times k) := ((a+b)/(a-b) - u \times v^2) - \sin(u+v) \times \cos(u-v) + r(i-j \times k)$$

Hier volgt eerst informatie over de primitiva en hun waarden.

10.1 Getal

Volgens [38] mag een getal genoteerd worden als geheel getal of als geheel getal gevolgd door punt (deze punt heeft dan geen betekenis) of als geheel getal gevolgd door punt gevolgd door decimale fractie.

Voorbeelden: 23 123. 3.47 Niet toegestaan is .47

10.2 Variabele

Er zijn twee soorten variabelen: [GP] of [GP-rij]([uitdr]), de laatste noemen we ook wel rij-element en de [uitdr] achter de [GP-rij] de index-expressie.

De waarde van een variabele is altijd de waarde die er bij executie het laatst aan is toegekend en die waarde is altijd een getal. Zoals bekend heeft een variabele behalve een waarde ook een adres.

Een GP-rij heeft evenveel adressen (en dus waarden) als is gegeven door ondergrens en bovengrens (zie 6). Het adres van een rij-element volgt uit de index, dat is de afgeronde waarde van de index-expressie. Er wordt gecontroleerd of deze index niet onder de ondergrens, of boven de bovengrens ligt; in dat geval volgt een foutmelding en wordt het programma afgevoerd. De waarde van een rij-element is het getal op het aldus aangewezen adres.

Voorbeeld

```
a := 2.3
b := 2.6
c := WRTL(2)      ; WRTL betekent vierkantswortel
r(a) := r(b) + r(c) ; dat is nu hetzelfde als : r(2) := r(3) + r(1)
```

Aan de schrijvers van compilers wordt aanbevolen ervoor te zorgen dat bij executie een foutmelding gegeven wordt zodra de waarde van een variabele gebruikt wordt voordat deze variabele in de loop van dit programma een waarde heeft gekregen.

10.3 FUN-roep

Bij een FUN-roep (zie ook hoofdstuk V) wordt het programma van de FUN-beschrijving doorlopen met de meegegeven aktuele(n). Een FUN-beschrijving eindigt altijd met een opdracht "FUN :=[uitdr] ". De waarde van [uitdr] is tevens de waarde die de FUN -roep aflevert.

10.4 Waarden van uitdrukkingen; prioriteiten

Wanneer de waarden der primitiva bekend zijn volgt de waarde van de uitdrukking door de operatoren + - x / M in de goede volgorde uit te voeren. Wat die goede volgorde is volgt uit onderstaande prioriteitsregels (die behalve ten aanzien van de deling overeenstemmen met Meneer Van Dalen Wacht Op Antwoord).

Omdat uitdrukkingen in een [vergelijking] (zie [50]) ook nog voorkomen in combinatie met vergelijkers, EN en OF geven we hier meteen een overzicht van al deze tekens in volgorde van hoogste naar laagste prioriteit; wat op dezelfde regel staat heeft gelijke prioriteit.

```
M
x /
+ -
< < = ≠ > >
EN
OF
```

Gelijke prioriteit betekent dat de operatoren in tekstuele volgorde aan de beurt komen.

Voorbeelden van gelijke prioriteiten

```
a - b + c      betekent  (a - b) + c
a / b x c      betekent  (a / b) x c   N.B.  Wijkt af van Nederlands gebruik.
a M b M c      betekent  (a M b) M c
```

Hogere prioriteit betekent dat een operator die tekstueel later komt dan een van lagere prioriteit, toch eerder wordt uitgevoerd.

Voorbeeld van verschillende prioriteiten

```
ALS a+bxc < pMq OF x = y-z EN u ≥ v/w DAN 12 ANDERS 13
betekent
```

```
ALS ((a+(bxc)) < (pMq)) OF ((x = (y-z)) EN (u ≥ (v/w))) DAN 12 ANDERS 13
```

Haakjesparen doorbreken uiteraard de prioriteitsregels: wat tussen haakjes staat moet eerst worden gedaan. Zodoende betekent $a + b \times c$ iets anders dan $(a + b) \times c$.

10.5 Volgorde bepaling primitiva

Met de prioriteitsregels alleen is men er nog niet. Zoals het voorbeeld $u := \text{LEES} / \text{LEES}$ laat zien kan de uitkomst nog afhangen van de volgorde waarin men de waarden van de primitiva bepaalt: als men eerst de eerste LEES uitvoert en daarna de tweede dan gebeurt er wat anders dan wanneer men eerst de tweede en daarna pas de eerste LEES uitvoert. Iets soortgelijks kan zich voordoen wanneer men FUN-roepen hanteert die de waarden der meegegeven [aktuele]n veranderen.

Afspraak

De primitiva in een [uitdr] worden van links naar rechts geëvalueerd. Bij een RIJ-reservering wordt eerst de [uitdr] voor de ondergrens en daarna die voor de bovengrens geëvalueerd.

Bij een [vergelijking] wordt eerst de eerste [uitdr] geëvalueerd, dan de tweede, dan (als er nog meer zijn, nl. bij het gebruik van EN en OF) de derde, etc.

10.6 Waarde van aMb

De waarde van aMb zal over het algemeen niet exact zijn. Voor niet te grote gehele waarden van a en niet-negatieve gehele waarden van b zal de uitkomst echter wel exact zijn.

11. Toekenning

Een toekenning (zie [30]) heeft de gedaante [variabele]:=[uitdr]. Bij executie wordt eerst het adres van de variabele in het linkerlid bepaald (het doeladres), daarna wordt de waarde van de uitdrukking in het rechterlid bepaald. Deze laatste waarde wordt opgeborgen in de door het doeladres aangewezen GP.

Voorbeeld

```
een := twee + drie
```

Het adres van de GP "een" wordt klaargezet als doeladres. De waarde van "twee" wordt opgehaald, de waarde van "drie" wordt daarbij opgeteld. Deze waarde wordt de nieuwe waarde van "een".

Voorbeeld

```
r(i+j×k) := a + b × WRTL(xM2 + yM2)
```

Eerst wordt de index-expressie $i+j \times k$ uitgerekend, de opgeleverde waarde wordt afgerond, dit is de index;

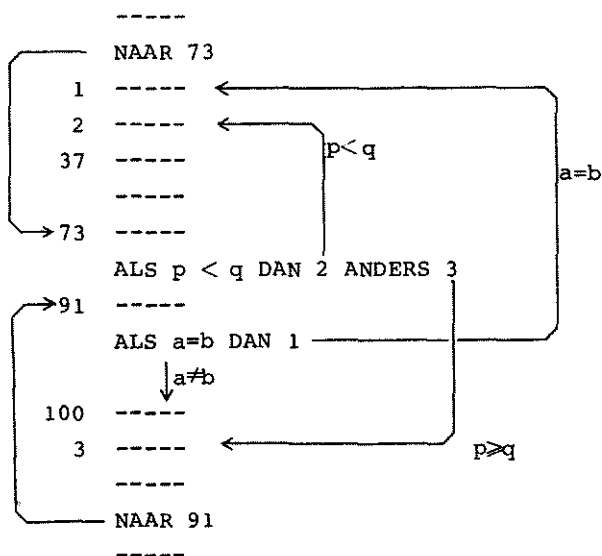
de index wordt op de grenzen uit de descriptor van "r" gecontroleerd, uit de index en het begin-adres van de rij wordt nu het adres berekend, het aldus berekende doeladres wordt klaargezet.

De waarde van het rechterlid wordt berekend uit de gegeven formule.

Deze waarde wordt opgeborgen in de GP waarvan het adres was klaargezet.

12. Sprong-opdrachten

Sprong-opdrachten dienen om bij executie een andere dan de tekstueel volgende aktie aan te wijzen als de vervolgens te executeren aktie. Een en ander blijkt voldoende uit onderstaand voorbeeld:



Men ziet dat de opdracht "ALS a=b DAN 1" hier hetzelfde doet als de opdracht "ALS a=b DAN 1 ANDERS 100" gedaan zou hebben.

Derhalve: bij een sprongopdracht wordt eerst de vergelijking achter ALS uitgewerkt. Zo'n vergelijking is WAAR of ONWAAR. Zie voor de betekenis van EN en OF onderstaand tabelletje.

De DAN-sprong wordt gedaan als de vergelijking WAAR is. Als de vergelijking ONWAAR is en er staat een ANDERS-sprong geschreven, dan wordt deze uitgevoerd, anders is de tekstueel volgende actie de vervolgens geëxecuteerde.

We brengen nog in herinnering (zie C7.4) dat wel naar allerlei acties (dus o.a. naar een RIJ-reservering) gesprongen mag worden maar niet naar START of een SUB- of FUN-beschrijving.

P	q	p EN q	p OF q
W	W	W	W
W	O	O	W
O	W	O	W
O	O	O	O

"W" is "WAAR"

"O" is "ONWAAR"

Voor een belangrijk neveneffect van sprongopdrachten zie 13 (het ongereserveerd maken van rijen).

13. RIJ-reservering

De executie van een RIJ-reservering betekent dat de waarden van onder en bovengrens worden uitgerekend, afgerond, en in de descriptor (zie 6) worden ingevuld. De aldus verkregen waarde van de bovengrens mag niet kleiner zijn dan die van de ondergrens, anders volgt een foutmelding. Vervolgens wordt in het geheugen van de computer het verlangde aantal GP's gereserveerd:

m := LEES

n := LEES

RIJ(m:n)r ; in orde als $m \leq n$, anders FOUT-melding, er wordt ruimte gereserveerd voor
; n - m + 1 GP's

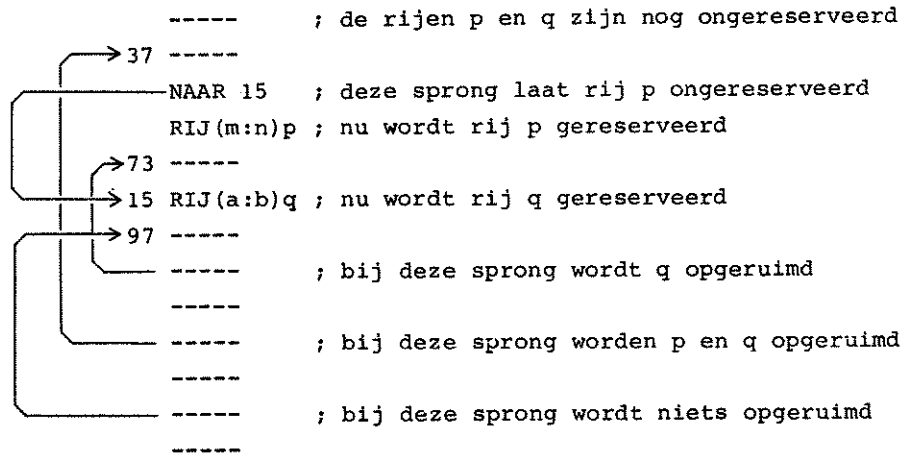
Men zie in dit verband nog de opmerkingen over volgorde in 10.6.

In plaats van RIJ(m:n)r mag men ook schrijven RIJSYM(m:n)r. Het enige verschil is dat men dan in de elementen van de rij uitsluitend gehele getallen tussen 0 en 99 (0 en 99 inbegrepen) mag onderbrengen. Deze faciliteit kan nuttig zijn bij het werken met zeer lange rijen kleine getallen, omdat de computer dergelijke RIJSYM's zuiniger in het geheugen kan opbergen dan gewone RIJen. Als men een waarde wil toekennen aan een element van een RIJSYM dan wordt deze waarde eerst afgerond. Als de afgeronde waarde niet een der getallen 0 t/m 99 is volgt een foutmelding, en wordt de executie gestaakt.

In verband met sprong-opdrachten is het volgende van veel belang:

- Voordat een programma wordt geëxecuteerd wordt ervoor gezorgd dat in de descriptor van a l l e rijen is aangetekend, dat deze "ongereserveerd" zijn.
- Wanneer men door een of andere actie een waarde wil toekennen aan of tevoorschijn halen uit een "ongereserveerde" rij, volgt o p d a t m o m e n t tijdens executie een fout-melding.
- Bij een "terugspringende" sprongopdracht (dat is een sprongopdracht die verwijst naar een plaats eerder in de programmatekst dan waar de sprongopdracht zelf staat) worden alle rijen, waarvan de reservering voorkomt in het stuk programmatekst waarover werd teruggesprongen, "ongereserveerd" verklaard en opgeruimd, d.w.z. de waarden van de elementen gaan verloren, en komen bij een later opnieuw reserveren van deze RIJ niet weer terug. (N.B. Als men "vooruit" springt over een RIJ-reservering dan was deze rij op grond van regels a en c al ongereserveerd vóór de sprong, en is dat nog steeds na de sprong omdat de betreffende reservering niet is uitgevoerd).

Voorbeeld



14. Cyclus

Volgens de syntax-regel [60] heeft een cyclus de gedaante

```

MET [stap], i := [start], [stop]
[programmeerdeel]n
[nummer] HERHAAL
  
```

(uiteraard mag bij het gebruik van de cyclus in een programmatekst ook voor MET een label staan, terwijl in [programmeerdeel] het mogen plaatsen van een label al besloten ligt, zie [21], [23] en [71]).

Hierin is [stap], [start] en [stop] ieder voor zich een [uitdr], dat wil dus zeggen: mogelijk een gecompliceerde uitdrukking met veel operanden, maar in de praktijk waarschijnlijk meestal niet gecompliceerder dan een [getal] of een heel simpele [formule]. Verder is "i" een [GP].

Losjes gezegd is de betekenis van de [cyclus] dat [programmeerdeel]ⁿ een aantal malen wordt uitgevoerd met telkens een andere waarde van i, te beginnen met de waarde van [start], oplopend met de waarde van [stap] en eindigend bij de waarde van [stop].

Voorbeelden

MET 2, i := 1, 7	}	[programmeerdeel] ⁿ	wordt uitgevoerd met achtereenvolgens i = 1,3,5,7
MET 2, i := 1, 8			
MET 2, i := 1, 1		[programmeerdeel] ⁿ	wordt uitgevoerd met i = 1
MET 2, i := 1, 0		[programmeerdeel] ⁿ	wordt helemaal niet uitgevoerd
MET-2, i := 3, -3	}	[programmeerdeel] ⁿ	wordt uitgevoerd met achtereenvolgens i = 3,1,-1,-3
MET-2, i := 3, -4			
MET-2, i := 3, 4		[programmeerdeel] ⁿ	wordt helemaal niet uitgevoerd

Als de cyclus verlaten wordt heeft i de waarde waarmee de cyclus voor het laatst doorlopen is, resp. de waarde van [start] als [programmeerdeel]ⁿ helemaal niet uitgevoerd is.

Zelfs bij deze eenvoudige voorbeelden moet nog een voorbehoud worden gemaakt: als men namelijk binnen [programmeerdeel]ⁿ de waarde van i verandert kan er best iets heel anders gebeuren.

Daarom zullen we nu precies zeggen wat er in een cyclus gebeurt. De cyclus is gelijkwaardig met het volgende stuk ECOL-programma. Hierin zijn de onderstreepte namen teller, stap en stop hulpvariabelen die door de compiler worden toegevoegd en waar de programmeur niet bij kan komen, en evenzo zijn label1, label2 en label3 labels die de programmeur niet kan bereiken.

Met dit stukje programma worden de kwadraten van de getallen 0 t/m 99 afgedrukt, telkens met tien op een regel. De vraag welke HERHAAL bij welke MET hoort is eenvoudig te beantwoorden: als twee MET's op elkaar volgen zonder HERHAAL ertussen komt de HERHAAL van de tweede MET vóór de HERHAAL van de eerste MET (vergelijk de haakjes structuur bij [uitdr]ukkingen).

In een cyclus mogen ook RIJ-reserveringen en (alhoewel dit geen bijzondere effecten heeft) ook SUB-beschrijvingen voorkomen. Dit is echter verboden bij het gebruik in een FUN-beschrijving, omdat daar geen RIJ-reserveringen en SUB-beschrijvingen mogen voorkomen.

15. Lege opdracht

Volgens [23] mogen in een ECOL tekst ook regels voorkomen waarop behalve eventueel een label en kommentaar verder niets staat, zulks ter bevordering van de leesbaarheid voor de menselijke lezer. Een dergelijke regel wordt bij executie gewoon overgeslagen, ook als men er via een sprong op terecht komt. Het is dus net of er op zo'n regel een opdracht staat die niets doet, een lege opdracht.

16. Invoer

```
16.1      [variabele] := LEES
          [variabele] := LEESYM
```

Met LEES wordt het eerstvolgende getal van een sequentieel extern medium gelezen, bijv. een ponsband of schrapkaart, en dit getal wordt als waarde toegekend aan de variabele in het linkerlid (de term sequentieel staat tegenover adresseerbaar; men kan niet zeggen: "ik wil de zoveelste hebben" maar slechts "geef mij de volgende"). Een getal op het externe medium moet een gedaante hebben als in syntax [38], eventueel voorafgegaan door + of - en ook hier geldt dat spaties geen betekenis hebben. LEES leest dan ook op het externe medium symbool na symbool totdat een cijfer verschijnt of een + of - gevolgd door een cijfer; hiermee is dan een getal begonnen, en dit getal wordt geacht op te houden zodra een symbool verschijnt dat niet meer tot het getal kan behoren, dus een symbool dat geen punt of cijfer is, of ook het symbool punt zelf als er in het getal al een punt is voorgekomen. Bij dit alles worden spaties steeds veronachtzaamd.

Het eerste symbool dat niet meer tot het getal behoort is tevens het eerste symbool dat door een volgende LEES of LEESYM gelezen wordt.

Als LEES tijdens het lezen van een getal bij het einde van het medium komt, is hiermee het getal ook afgelopen. Als LEES op het externe medium geen getal meer aantreft volgt een foutmelding, alsmede beëindiging der executie van het programma.

Met LEESYM wordt het eerstvolgende symbool van een sequentieel extern medium gelezen, en de interne waarde van dit symbool wordt aan de variabele in het linkerlid toegekend. De interne waarde van een extern symbool is een geheel getal. De relatie is in onderstaande tabel vastgelegd. Als LEESYM een ander symbool ontmoet dan de onder 0 t/m 91 genoemde, wordt de waarde 98 afgeleverd (zie os = onbekend symbool). Als het externe medium uitgeput is levert LEESYM de waarde 99 af (zie et = einde tekst).

Als na LEESYM een LEES wordt uitgevoerd is het eerste symbool dat LEES leest het laatste dat LEESYM gelezen heeft. Hierdoor is het mogelijk met LEESYM te speuren naar het begin van een nieuw getal en dan toch nog met LEES het hele getal te lezen.

TABEL van symbolen en hun interne waarde:

cijfers		letters								overige symbolen									
sym get		sym get		sym get		sym get		sym get		sym get		sym get		sym get					
0	0	a	10	k	20	u	30	A	40	K	50	U	60	.	70	+	80	sp	90
1	1	b	11	l	21	v	31	B	41	L	51	V	61	,	71	-	81	nr	91
2	2	c	12	m	22	w	32	C	42	M	52	W	62	:	72	x	82		
3	3	d	13	n	23	x	33	D	43	N	53	X	63	;	73	/	83		
4	4	e	14	o	24	y	34	E	44	O	54	Y	64	?	74	=	84		
5	5	f	15	p	25	z	35	F	45	P	55	Z	65	!	75	<	85		
6	6	g	16	q	26			G	46	Q	56			'	76	>	86		
7	7	h	17	r	27			H	47	R	57			"	77	≤	87		
8	8	i	18	s	28			I	48	S	58			(	78	≥	88	os	98
9	9	j	19	t	29			J	49	T	59			)	79	≠	89	et	99

Met sp en nr bedoelen we 'spatie' en 'overgang op nieuwe regel', met et bedoelen we 'einde tekst' (om aan te geven dat een tekst afgelopen is), met os een 'overig symbool' of 'onbekend symbool'.

Voorbeeld 16

Het externe medium bevat

+.- 5 . A+.5 7.23 . -17

en wordt gelezen met

```

a:= LEES      ; leest tot A, -5 naar a
b:= LEES      ; begint bij A en leest tot vijfde punt; 57.23 naar b
c:= LEESY     ; leest vijfde punt; 70 naar c
d:= LEESY     ; leest spatie; 90 naar d
e:= LEESY     ; leest -; 81 naar e
f:= LEES      ; begint bij - en leest tot einde; -17 naar f
g:= LEESY     ; medium uitgeput; 99 naar g

```

Tenslotte: men mag LEES en LEESY ook in [uitdr]ukkingen gebruiken (zie 18).

16.2

```

          h          k
          ↓          ↓
[GP-rij] := BESTAND([uitdr] , [uitdr] , [uitdr])

```

De eerste parameter "[uitdr]" is altijd het bestand-nummer (mogelijk is op sommige computers hier alleen maar een [getal] toegestaan i.p.v. een willekeurige [uitdr]).

De opdracht kopieert het stuk h t/m k van het aangewezen bestand in de rij met de naam [GP-rij].

Het kopiëren eindigt zodra of de rij, of het aangegeven stuk bestand uitgeput is.

Alle [uitdr] worden weer eerst afgerond.

16.3 [GP-rij] := BESTAND([uitdr])

doet hetzelfde als:

[GP-rij] := BESTAND([uitdr], 1, totale lengte van het bestand)

17. Uitvoer

17.1

```

          m          n
          ↓          ↓
SCHRIJF([uitdr] , [uitdr]) := [uitdr]

```

Het uitvoerapparaat (in het algemeen een regeldrukker) drukt de waarde van de [uitdr] rechts van het wordt-teken in decimale voorstelling af met teken (een plusteken wordt mogelijk vervangen door spatie), m cijfers voor de decimale punt en n cijfers erachter, in totaal dus m+n+2 posities. Bij

het gedeelte voor de punt worden niet-significante nullen onderdrukt. Als $n=0$ worden evenwel geen decimale punt met cijfers erachter afgedrukt, dus slechts $m+1$ posities. De [uitdr] voor m en n worden weer afgerond.

Wanneer het gedeelte vóór de decimale punt meer cijfers nodig heeft dan door m is aangegeven, of wanneer m te groot is (varieert van machine tot machine) dan wordt het getal afgedrukt volgens SCHRIJM.

17.2 SCHRIJF := [uitdr]

Als de waarde van [uitdr] geheel is doet deze opdracht hetzelfde als $SCHRIJF(6,0) := [uitdr]$, in het andere geval hetzelfde als $SCHRIJF(6,4) := [uitdr]$.

In beide gevallen worden zoveel spaties toegevoegd dat op het uitvoerapparaat evenveel posities worden bezet als met SCHRIJM.

17.3 SCHRIJM := [uitdr]

Het uitvoer-apparaat drukt de afgeronde waarde rechts van het ":" teken in decimale voorstelling af als een mantisse tussen 0 en 1 gevolgd door de exponent van de macht van 10 waarmee dit getal vermenigvuldigd moet worden om de waarde van de meegegeven [uitdr] voor te stellen. Zowel mantisse als exponent worden voorafgegaan door een voorteken.

Het aantal decimalen van mantisse en exponent is afhankelijk van de machine waarop het programma wordt verwerkt.

Voorbeeld

SCHRIJF := $\pi \times 10^5$	⇒	314159.2654
SCHRIJF(3,5) := $\pi \times 10^5$	⇒	314.15927
SCHRIJF := $-\pi \times 10^5$	⇒	- 3141.5927
SCHRIJM := $\pi \times 10^5$	⇒	+0.3141592653589E+6
SCHRIJF(5,5) := $\pi \times 10^5$	⇒	+0.3141592653589E+6
SCHRIJF := -12	⇒	- 12

17.4 SCHRIJFSYM := [uitdr]

Er wordt gekeken of de op een geheel getal afgeronde waarde van [uitdr] hoogstens 91 is en voorkomt in de tabel van symbolen (zie 16.1). Zo ja, dan wordt het symbool met die waarde afgedrukt; zo nee, dan wordt een bijzonder symbool afgedrukt. Wat er op papier verschijnt als het uitvoer-apparaat een symbool moet afdrukken waartoe het niet in staat is (bijv. een kleine letter als het alleen hoofdletters kent) kan van machine tot machine verschillen.

Voorbeeld

SCHRIJFSYM := 74 drukt af ?

17.5 TEKST := "[sym]ⁿ⁺¹"

Afgedrukt wordt de tekst tussen de aanhalingstekens; in deze tekst mag echter geen aanhalingsteken voorkomen en geen [⌋].

17.6 SPATIE([uitdr])

Het uitvoer-apparaat geeft net zoveel spaties als door de afgeronde waarde van [uitdr] wordt aangegeven; als deze afgeronde waarde ≤ 0 is, gebeurt er niets.

17.7 SPATIE

doet hetzelfde als de opdracht:

SPATIE(1)

17.8 NR([uitdr])

Het uitvoer-apparaat schuift het papier evenveel regels omhoog als door de afgeronde waarde van [uitdr] wordt aangegeven; als deze afgeronde waarde ≤ 0 is, gebeurt er niets met het papier. Na het omhoogschuiven, maar ook bij "NR(0)" begint het afdrukken weer vooraan de regel (dit vooraan laten

beginnen bij NR(0) kan bij sommige machines op moeilijkheden stuiten).

17.9 NR

doet hetzelfde als de opdracht

NR(1)

17.10 NP

Deze opdracht laat op het uitvoerapparaat een nieuwe pagina verschijnen.

17.11

h k
↓ ↓
BESTAND([uitdr] , [uitdr] , [uitdr]) := [GP-rij]

De eerste parameter "[uitdr]" is altijd het bestand-nummer (zie opmerking in 16.2).

De opdracht kopieert de rij met de naam [GP-rij] in het stuk h t/m k van het aangewezen bestand. Het kopiëren eindigt zodra of de rij, of het aangegeven stuk bestand uitgeput is.

17.12 BESTAND([uitdr]) := [GP-rij]

doet hetzelfde als de opdracht:

BESTAND([uitdr] , 1, totale lengte van het bestand) := [GP-rij]

17.13 Opmerkingen uitvoer

De wijze waarop de computer reageert op situaties zoals:

- men wil meer symbolen op een regel hebben dan mogelijk is bij een bepaald uitvoerapparaat,
- men wil een symbool afdrukken dat er niet is op het uitvoerapparaat

kan van machine tot machine verschillen, evenals de voorstelling van allerlei symbolen en tekens (bijv. "*" in plaats van "x", "<=" in plaats van "≤", "10" in plaats van "E" in een SCHRIJMGetal).

Men bedenke hierbij ook, dat vele computers regeldrukkers van vaak uiteenlopend type hebben, en het lang niet zeker is dat de ECOL-uitvoer altijd over hetzelfde type regeldrukker loopt.

18. Standaard-functies

De volgende standaard-functies worden in iedere ECOL-implementatie verondersteld: ("u" staat hier steeds voor een [uitdr], tenzij anders vermeld):

G(u)	levert het gehele deel van u af:	G(2.7) is +2 G(-2.3) is -3
ABS(u)	levert de absolute waarde van u af	
WRTL(u)	levert $\sqrt{u}$ af als $u \geq 0$; anders foutmelding	
SQRT(u)	doet hetzelfde als WRTL(u)	
SIN(u)	levert de waarde van de sinus van u (in radialen) af	
COS(u)	levert de waarde van de cosinus van u (in radialen) af	
ARCTAN(u)	levert de waarde van de hoek (in radialen) af die gelegen is tussen $-\pi/2$ en $+\pi/2$ en waarvan de tangens u is	
LN(u)	levert $e \log(u)$ af als $u > 0$; anders foutmelding	
LOG(u)	levert $10 \log(u)$ af als $u > 0$; anders foutmelding	
EXP(u)	levert de waarde van e tot-de-macht u af	
GOK(u1,u2)	levert (pseudo) aselechte getallen af, t.w. met ongeveer gelijke waarschijnlijkheid een der gehele getallen tussen de afgeronde waarden van u1 en u2, deze afgeronde waarden inbegrepen. Dus GOK(1,6) en GOK(6,1) werken als dobbelsteen	

GOKC levert (pseudo) aselechte getallen af, t.w. met ongeveer gelijke waarschijnlijkheid een der getallen tussen 0 en 1, 0 wel en 1 niet inbegrepen

ONDIN([GP-rij]) levert de ONDerINdex af van de rij met naam [GP-rij]

BOVIN([GP-rij]) " " BOVenINdex af van " " " " [GP-rij]

LEES leest het volgende getal van de band en levert daarvan de waarde af net zoals in de invoer-opdracht [variabele] := LEES. Men mag dus ook schrijven bijv. "y := LEES + 1"

LEESSYM idem maar nu voor het volgende symbool op de band of kaart

SKIP doet net zo vaak LEESSYM totdat [↵] is gepasseerd

PS (positie schrijfkop) levert het aantal symbolen af dat door het uitvoerapparaat op de huidige regel is afgedrukt

PP (positie papier) levert het aantal voltooide regels op de huidige pagina van het uitvoer-apparaat

STOP de executie van dit programma wordt beëindigd

V. SUB EN FUN

19. Inleiding

Het is bij het schrijven van programma's niet ongebruikelijk dat men in verschillende delen van zijn programma gelijke of bijna gelijke reeksen opdrachten nodig heeft. Het SUB en FUN mechanisme maakt het nu mogelijk zo'n reeks opdrachten eenmaal op te schrijven, waarna ze van verschillende plaatsen door één simpele opdracht kunnen worden geactiveerd. Men zie bijv. de standaardfuncties (zie 18), die eenmaal ergens in het geheugen van de computer uitgeprogrammeerd staan, en die men in zijn programma overal heel simpel kan aanroepen. De SUB is een heel eenvoudig mechanisme, dat dan ook niet zo erg flexibel is, de FUN opent veel verder gaande mogelijkheden.

20. SUB-beschrijving

Volgens syntax-regel [71] is een [SUB-beschrijving] :

```
[nummer] SUB[ SUB-naam] [
[opdracht]n
[nummer] END [
```

In een SUB-beschrijving mogen dus geen RIJ-reserveringen en ook geen SUB-beschrijvingen voorkomen. Wel zijn alle [opdrachten] , inclusief de [cyclus] en de [SUB-roep] , toegestaan.

Omdat een SUB-beschrijving deel uitmaakt van het hoofdprogramma, gelden in het bijzonder de in 7 en 8 behandelde context-regels.

Men noemt een SUB-beschrijving ook wel een subroutine.

Voorbeeld van een SUB-beschrijving

```
100 SUB skip tot nul                    ; deze subroutine leest de getalband tot en met het eerst-
                                         ; volgende getal 0

101 getal := LEES

102 ALS getal ≠ 0 DAN 101

103 END                                  ; de subroutine wordt verlaten via de achterdeur, zoals het be-
                                         ; hoort, zie 8.
```

Deze SUB-beschrijving mag overal in het hoofdprogramma staan mits buiten een SUB-beschrijving, en ook overal in het hoofdprogramma worden aangeroepen (dus wel in een SUB-beschrijving, niet in een FUN-beschrijving).

variabelen) maar ook RIJ-namen. In de FUN-beschrijving moet men aangeven wat voor soort aktuelen men wenst mee te geven.

Voorbeeld van FUN-beschrijving

```
FUN som(h,k,RIJ a)
  hulp := 0
  MET l, i:=h, k
  hulp := hulp + a(i)
  HERHAAL
  FUN := hulp
```

Deze FUN bepaalt voor een meegegeven RIJ, die we hier maar zolang a noemen ("formele" naam) de som van het h-de tot en met het k-de element (h en k zijn ook formelen). De aanduiding RIJ vóór a geeft aan dat we bij de aanroep als derde parameter de naam van een rij mee moeten geven. Het ontbreken van deze aanduiding bij h en k geeft aan dat men bij de aanroep op de eerste en tweede plaats een uitdrukking moet zetten. Nu een

Voorbeeld van aanroep van deze FUN vanuit het hoofdprogramma

```
-----
-----
RIJ(1:500)m
-----
-----
p := som(6, 34, m)
q := som(3, m(6), m)
```

Met de eerste aanroep worden de elementen 6 t/m 34 van de rij m opgeteld, en de som wordt aan p als waarde toegekend. Bij de tweede aanroep wordt de som van de elementen 3 t/m m(6) aan q toegekend. Dat m, een rijnaam, ook nog eens op de plaats van de tweede parameter voorkomt, terwijl de tweede parameterplaats in de FUN-beschrijving niet met RIJ is gemerkt, is niet onjuist: immers de tweede aktuele luidt m(6), dat is dus gewoon een variabele, dus een uitdrukking.

Een heel belangrijke, en ook doorgaans prettige, eigenschap van de FUN is dat alle er in optredende labels en namen (behalve de namen van andere FUN's) lokaal zijn, d.w.z. niets te maken hebben met identieke labels en namen die mogelijk in andere FUN's of in het hoofdprogramma gebruikt worden. En in een bepaalde FUN-beschrijving hoeft men met de namen van andere FUN's alleen nog maar rekening te houden voor zover men die andere FUN's vanuit deze FUN-beschrijving wenst aan te roepen. De namen van FUN's die men niet wenst aan te roepen mag men rustig als (lokale) GP's of formelen gebruiken. Bij het schrijven van een FUN kan men dus de te gebruiken labels en namen (behalve namen van aan te roepen FUN's) vrij kiezen, ongeacht welke namen in andere FUN-beschrijvingen of in het hoofdprogramma gebruikt worden. Dit maakt het o.a. mogelijk voor vaak voorkomende ingewikkelde processen (bijv. het oplossen van stelsels vergelijkingen) FUN-beschrijvingen te maken die men aan elk willekeurig programma kan toevoegen.

23. FUN-beschrijving

Volgens [86] t/m [89] begint een FUN-beschrijving met FUN gevolgd door de naam die men aan de FUN wenst te geven, gevolgd door een rijtje formelen, minstens één, van elkaar gescheiden door komma's dit rijtje gevat in haakjes. Elke formele heeft de gedaante [GP] (d.w.z. [naam]) of RIJ [GP-rij] (d.w.z. het woordje RIJ gevolgd door een [naam]).

Na deze "kop" van de FUN-beschrijving komen [opdrachten], d.w.z. dat RIJ-reserveringen en SUB-beschrijvingen niet zijn toegestaan. Maar ook zijn SUB-roepen niet toegestaan, zoals aanstonds zal blijken.

De FUN-beschrijving eindigt tenslotte met FUN := [uitdr] (en niet met END, zoals de SUB-beschrijving).

Een zelfgeschreven FUN moet dus, in afwijking van de standaardfuncties, minstens één parameter hebben, ook al heeft men die eigenlijk helemaal niet nodig. Men is evenwel niet verplicht deze

parameter in de rest van de FUN-beschrijving ook te gebruiken. Overigens zou men met parameterloze, procedures ook niet erg veel kunnen aanvangen, enkele bijzondere gevallen daargelaten. Evenzo moet een FUN-beschrijving eindigen met FUN :---, ook al is men helemaal niet van plan van de aldus toegekende waarde gebruik te maken, bijv. doordat de FUN-roepen van deze FUN nooit in een [uitdr] maar altijd als [opdracht] plaats hebben (zie [23] en [37]).

De verschillende [FUN-beschrijvingen] zijn dermate onafhankelijk van elkaar dat men voor elke [FUN-beschrijving] apart bij het in tekstuele volgorde lezen weer zekerheid heeft over de status van een [naam] bij het eerste optreden ervan.

Context-regels voor het funktiedeel

Voor elke [FUN-beschrijving] afzonderlijk geldt:

- C23.1 Een en dezelfde [naam] heeft overal in deze [FUN-beschrijving] dezelfde betekenis.
- C23.2 Een [naam] is een [GP-rij] als hij in deze [FUN-beschrijving] optreedt op de plaats van [formele] voorafgegaan door de aanduiding RIJ (zie [86] en [89]).
- C23.3 Een [naam] is een [FUN-naam] als hij in deze [FUN-beschrijving] voorkomt gevolgd door een openingshaakje (spaties spelen geen rol), en niet valt onder C23.2.
- C23.4 Alle overige namen zijn [GP]'s voor deze [FUN-beschrijving].
- C23.5 Alle [labels] in een [FUN-beschrijving] moeten van elkaar verschillen.
- C23.6 Alle [namen] in deze [FUN-beschrijving] die geen [FUN-naam] zijn, en alle [labels] hebben niets te maken met eventuele identieke [namen] en [labels] elders in het programma.

Voor het funktiedeel als geheel geldt:

- C23.7 Elke [naam] die in een of meer der [FUN-beschrijvingen] als [FUN-naam] optreedt moet ook precies éénmaal optreden op de plaats van [FUN-naam] in een [FUN-beschrijving].

Uit deze context-regels volgt nog dat SUB-roepen vanuit een FUN-beschrijving onmogelijk zijn, want SUB-namen zijn namen in het hoofdprogramma.

Voorbeeld

```

FUN f(x)      ; Hiermee is "f" voor het gehele programma, inclusief het funktiedeel, een
               ; [FUN-naam], "x" is, alleen voor deze FUN-beschrijving, een formele [GP]
y := x        ; Hiermee is "y", echter alleen voor deze FUN-beschrijving, een zg. "lokale"
               ; [GP]
1  x := r(y)   ; "r" kan onmogelijk de naam zijn van een [GP-rij] want dan zou "r" als zo-
               ; danig formeel moeten zijn. De compiler vertaalt "r" nu als de naam van een
               ; FUN met 1 formele [GP], waarvan de beschrijving nog moet volgen
               ; De label "1" is ook "lokaal", dat wil zeggen: geldt alleen voor deze FUN-
               ; beschrijving
FUN := x+y     ; Hiermee wordt deze FUN-beschrijving afgesloten
               ; De betekenis van de namen "x" en "y" en van de label "1" gaan nu verloren,
               ; n i e t echter de betekenis van de namen "f" en "r" als FUN-namen
               ; De compiler verwacht nog de beschrijving van "r".
FUN r(x,RIJp); FOUT! "r" mocht alleen maar 1 formele [GP] hebben
-----
-----
FUN := -----
FUN r(x)      ; Dat verwacht de compiler (zie overigens C23.7)
-----
-----
FUN :=-----

```

24. FUN-roep

De executie van een FUN-roep betekent, dat het programma van de aangeroepen FUN-beschrijving wordt doorlopen. Zodra daarbij "FUN :=[uitdr]" wordt bereikt, wordt teruggekeerd naar het punt vanwaar de funktie werd aangeroepen. Wanneer dit plaats vond in een [uitdr] , is de waarde van de FUN-roep, de aan "FUN" toegekende waarde; wanneer de funktie als [opdracht] werd geroepen, speelt deze waarde verder geen rol (zie voorbeeld 24.2).

In een FUN-roep moet op de plaats van een formele [GP] als [aktuele] een [uitdr] worden meegegeven en op de plaats van een formele [GP-rij] een (aktuele) [GP-rij] (niet voorafgegaan door de aanduiding RIJ). Deze rij dient ten tijde van de aanroep gereserveerd te zijn.

Als de aktuele [uitdr] een [variabele] is, wordt bij executie het adres van die [variabele] aan de funktie doorgegeven, anders wordt die [uitdr] uitgerekend en in een voor de programmeur onbereikbare hulp-GP gestopt waarvan bij executie ook het adres aan de funktie wordt meegegeven.

De waarden der bedoelde [uitdr]ukkingen en de adressen der [variabelen] worden weer bepaald in de tekstuele volgorde der parameters (vgl. 10.6).

Omdat bij executie het adres van de aktuele wordt meegegeven opereert de FUN ook echt op die aktuele. Wordt in een FUN-beschrijving een waarde toegekend aan een formele GP, dan wordt bij aanroep aan de aktuele GP een waarde toegekend. Zo kan de FUN, ondanks context-regel C23.6 toch via de parameters waarden van GP's in het hoofdprogramma veranderen.

Evenzo wordt bij een aktuele [GP-rij] de descriptor van de rij meegegeven, zodat weer op de echte rij-elementen wordt geopereerd.

Voorbeeld 24.1

```
FUN f(x,y)
  x := 3
  FUN := y
```

Met een aanroep f(a,b) krijgt a de waarde 3. Met een aanroep f(a+b,c) of f(17,q) krijgt de hulp-GP, waar de waarde van a+b resp 17 ingestopt was, de waarde 3, maar met a, b of overige getallen 17 in het programma gebeurt niets. Interessant is nog de aanroep f(i,d(i)). Als bij aanroep i bijv. de waarde 5 heeft, krijgt FUN de waarde d(5), en niet d(3), zoals men misschien zou denken (omdat i door FUN de waarde 3 krijgt). Immers bij aanroep wordt het adres van de parameters meegegeven, dus het adres van d(5).

Als de FUN-beschrijving daarentegen geluid had

```
FUN g(x,RIJ y)
  x := 3
  FUN := y(x)
```

dan zou bij aanroep met g(i,d) (d een op dat moment gereserveerde rij) FUN de waarde d(3) hebben gekregen.

Voorbeeld 24.2

```
FUN max(index,RIJ r)           ; deze funktie levert het grootste getal van de rij r en kent
                                ; aan "index" de waarde toe van de index waarvoor dat maximum
                                ; wordt bereikt
h := ONDIN(r)                  ; h := onderindex van r, zie 18
k := BOVIN(r)                  ; k := bovenindex van r, zie 18
index := h
probeer := r(h)
MET 1, i := h+1, k
  volgende := r(i)
  ALS volgende ≤ probeer DAN 1
    index := i
    probeer := r(i)
1 HERHAAL
FUN := probeer
```

Als "max" wordt aangeroepen bijvoorbeeld in:

```
y := max(i, rij)
```

wordt aan "y" de waarde van het maximale element van de aktuele "rij" toegekend, terwijl de index waarvoor dit maximum wordt bereikt in "i" is te vinden.

Deze "i" uit de aanroep (een GP uit het hoofdprogramma, of een andere FUN-beschrijving!) heeft niets te maken met de "i" van de beschrijving "max".

Als "max" wordt aangeroepen als [opdracht]:

```
max(u,rij)
```

gooien we de waarde van het maximale getal van "rij" weg, maar die kunnen we altijd weer terugvinden door de opdracht:

```
grootste := rij(u)
```

Als "max" wordt aangeroepen bijvoorbeeld als in:

```
y := max(0,rij)
```

wordt de aktuele waarde 0 in een voor de programmeur onbereikbare GP gestopt. Deze onbereikbare GP gaat bij het doorlopen van "max" nu fungeren als hulp-GP voor de formele "index", maar de laatste waarde daarvan ben je wel kwijt als de funktie klaar is!

25. Recursiviteit

Funkties mogen ook "in recursie" gaan, dat wil zeggen: zichzelf (hetzij direkt of via een andere funktie) aanroepen. Bij recursief gebruik kan een funktie dus in verschillende "incarnaties" (of "diepten") worden geactiveerd. Er komt dan nog een regel bij:

C25.1 Als een funktie in recursie wordt aangeroepen, worden voor alle lokale-, niet-formele [GP]'s, nieuwe geheugenplaatsen gereserveerd, die niets te maken hebben met de geheugenplaatsen die voor die [GP]'s in enige vorige incarnatie waren gereserveerd.

Dit recursie mechanisme is een afspiegeling van en kan gebruikt worden bij rekenmachine implementatie voor recursieve processen in de wiskunde. Een heel bekend voorbeeld van zo'n recursief wiskundig proces is de algoritme van Euclides voor het bepalen van de grootste gemene deler van twee (niet noodzakelijk positieve) gehele getallen m en n. In woorden luidt deze algoritme:

```
als n = 0 dan is de ggd van m en n het getal m
als n ≠ 0 dan is de ggd van m en n hetzelfde als de ggd van n en de rest die ontstaat
bij deling van m door n.
```

Men vindt dit vrijwel letterlijk terug in onderstaande FUN-beschrijving. Alleen, omdat men in een FUN-beschrijving niet tweemaal mag zeggen FUN :=--- vindt men op regel 2 een toekenning aan m, die op dat moment toch overbodig is en als hulpvariabele dienst doet.

```
FUN ggd(m,n)
1 ALS n = 0 DAN 3
2 m := ggd(n,m-n × G(m/n)) ; m - n×G(m/n) is rest bij deling van m door n
3 FUN := ABS(m) ; we maken de ggd positief
```

VI DE ECOL-SYNTAX

In de syntax wordt min of meer vastgelegd welke teksten wel, en welke niet door de compiler als correct ECOL worden geaccepteerd. Hierbij maken we gebruik van zg. "produktie-regels", zoals [5], waarvan de betekenis geen toelichting zal behoeven, en waarbij een ieder zich wel kan voorstellen welke letters niet expliciet opgeschreven zijn.

In [8] is ruimte gelaten voor eventueel nog enkele leestekens die mogelijk niet op elk apparaat (of schrapkaart) produceerbaar zijn; de gegeven leestekens zijn de minimaal vereiste.

Vele produktieregels produceren een opeenvolging van syntactische eenheden, zoals [55]. Men moet dit zo lezen dat een [RIJ-reservering] bestaat uit het woordje RIJ, gevolgd door een openingshaakje, waarna iets komt dat door de produktieregel voor [uitdr] (zie [34]) geproduceerd wordt, waarna dubbelpunt etc.

In een aantal produktieregels komt een "exponent" n of $n+1$ voor. Bijvoorbeeld in [12], waar dit aanduidt dat een [naam] een opeenvolging is van $n+1$ [kleine letter]s. Steeds is $n = 0, 1, 2, 3, \dots$. Een [naam] bestaat dus uit minstens één kleine letter.

Evenzo betekent [opdracht]ⁿ (zie [86]) dat er n [opdracht]en na elkaar mogen komen, maar omdat n ook 0 mag zijn mag er ook helemaal niks staan. Als er meer dan één [opdracht] staat moeten deze vanzelf onder elkaar komen te staan, omdat krachtens [23] elke opdracht wordt beëindigd met [] (zie hiervoor [16]).

In enkele gevallen tenslotte is een produktieregel recursief, zoals bijv. [88]; men moet dit zo lezen:

[formelen] is [formeel] of [formeel],[formeel]
of [formeel],[formeel],[formeel] etc

Bij de recursie kunnen meerdere produktieregels betrokken zijn, zoals het geval is in [34] t/m [37].

Op vanzelfsprekende wijze worden in de hoofdstukken I t/m V van dit rapport verwijzingen naar syntactische begrippen ook in de meervoudsvorm gegeven, dus [namen],[labels], e.d.

Voor apparaten met een kleinere symbolen-voorraad dan minimaal voor ECOL vereist (bijv. geen onderscheid tussen [kleine letter] en [HOOFDLETTER]), kunnen eenvoudige conventies worden opgesteld, die het toch mogelijk maken ECOL-programma's via een dergelijk apparaat in te voeren. Deze conventies zijn echter veelal machine-, althans implementatie-gebonden en worden hier verder buiten beschouwing gelaten.

Spatieconventie

In de syntax is eenvoudigheidshalve niet tot uitdrukking gebracht dat overal in een [ECOL-programma] spaties mogen worden toegevoegd zonder dat daardoor de betekenis van het programma verandert, behalve in een [hoofdletterwoord] (= [HOOFDLETTER]ⁿ⁺¹) en tussen de aanhalingstekens achter TEKST:=. Er moet echter minstens één spatie staan wanneer twee [hoofdletterwoord]en direct op elkaar volgen.

[1] [sym]	<u>is</u>	[⤵] <u>of</u> [symbool]	
[2] [⤵]	<u>is</u>	[overgang op nieuwe regel, of einde kaart]	
[3] [symbool]	<u>is</u>	[cijfer] <u>of</u> [kleine letter] <u>of</u> [HOOFDLETTER] <u>of</u> [teken] <u>of</u> [spatie]	
[4] [cijfer]	<u>is</u>	0 <u>of</u> 1 <u>of</u> 2 <u>of</u> 3 <u>of</u> 4 <u>of</u> 5 <u>of</u> 6 <u>of</u> 7 <u>of</u> 8 <u>of</u> 9	
[5] [kleine letter]	<u>is</u>	a <u>of</u> b <u>of</u> c <u>of</u>	<u>of</u> x <u>of</u> y <u>of</u> z
[6] [HOOFDLETTER]	<u>is</u>	A <u>of</u> B <u>of</u> C <u>of</u>	<u>of</u> X <u>of</u> Y <u>of</u> Z
[7] [teken]	<u>is</u>	[leesteken] <u>of</u> [operator] <u>of</u> [vergelijker]	
[8] [leesteken]	<u>is</u>	. <u>of</u> , <u>of</u> : <u>of</u> ; <u>of</u> ? <u>of</u> ! <u>of</u> (<u>of</u>) <u>of</u>	<u>of</u> "
[9] [operator]	<u>is</u>	+ <u>of</u> - <u>of</u> × <u>of</u> / <u>of</u> M	
[10] [vergelijker]	<u>is</u>	= <u>of</u> < <u>of</u> > <u>of</u> ≤ <u>of</u> ≥ <u>of</u> ≠	

[11] [niks]	<u>is</u>	
[12] [naam]	<u>is</u>	[kleine letter] ⁿ⁺¹
[13] [NAAM]	<u>is</u>	[HOOFDLETTER] ⁿ⁺¹
[14] [label]	<u>is</u>	[cijfer] ⁿ⁺¹
[15] [nummer]	<u>is</u>	[label] <u>of</u> [label]: <u>of</u> [niks]
[16] []	<u>is</u>	[↻] <u>of</u> ; [symbool] ⁿ [↻]

[20] [ECOL-programma]	<u>is</u>	[FUN-beschrijving] ⁿ [nummer] START [] [programmadeel] ⁿ [nummer] KLAAR []
[21] [programmadeel]	<u>is</u>	[actie] <u>of</u> [SUB-beschrijving]
[22] [actie]	<u>is</u>	[opdracht] <u>of</u> [RIJ-reservering]
[23] [opdracht]	<u>is</u>	[nummer][toekenning] [] <u>of</u> [nummer][invoer/uitvoer] [] <u>of</u> [nummer][sprong] [] <u>of</u> [nummer][cyclus] [] <u>of</u> [nummer][SUB-roep] [] <u>of</u> [nummer][FUN-roep] [] <u>of</u> [nummer] []
[30] [toekenning]	<u>is</u>	[variabele] := [uitdr]
[31] [variabele]	<u>is</u>	[GP] <u>of</u> [GP-rij] ([uitdr])
[32] [GP]	<u>is</u>	[naam]
[33] [GP-rij]	<u>is</u>	[naam]
[34] [uitdr]	<u>is</u>	[voorteken][formule]
[35] [voorteken]	<u>is</u>	[niks] <u>of</u> + <u>of</u> -
[36] [formule]	<u>is</u>	[operand] <u>of</u> [formule][operator][operand]
[37] [operand]	<u>is</u>	[getal] <u>of</u> [variabele] <u>of</u> [FUN-roep] <u>of</u> ([uitdr])
[38] [getal]	<u>is</u>	[cijfer] ⁿ⁺¹ <u>of</u> [cijfer] ⁿ⁺¹ , <u>of</u> [cijfer] ⁿ⁺¹ . [cijfer] ⁿ⁺¹
[40] [invoer/uitvoer]	<u>is</u>	[variabele] := LEES <u>of</u> [variabele] := LEESYM <u>of</u> [GP-rij] := BESTAND([uitdr]) <u>of</u> [GP-rij] := BESTAND([uitdr],[uitdr],[uitdr]) <u>of</u> SPATIE <u>of</u> SPATIE([uitdr]) <u>of</u> NR <u>of</u> NR([uitdr]) <u>of</u> NP <u>of</u> TEKST := "[symbool] ⁿ⁺¹ " <u>of</u> SCHRIJF := [uitdr] <u>of</u> SCHRIJM := [uitdr] <u>of</u> SCHRIJFSYM := [uitdr] <u>of</u> SCHRIJF([uitdr],[uitdr]) := [uitdr] <u>of</u> BESTAND([uitdr]) := [GP-rij] <u>of</u> BESTAND([uitdr],[uitdr],[uitdr]) := [GP-rij]

[45] [sprong]	<u>is</u>	NAAR [label] <u>of</u> ALS[vergelijking] DAN [label] ANDERS [label] <u>of</u> ALS[vergelijking] DAN [label]
[50] [vergelijking]	<u>is</u>	[uitdr][vergelijker][uitdr] <u>of</u> [vergelijking] EN [vergelijking] <u>of</u> [vergelijking] OF [vergelijking] <u>of</u> ([vergelijking])
[55] [RIJ-reservering]	<u>is</u>	RIJ ([uitdr] : [uitdr])[GP-rij] [] <u>of</u> RIJSYM([uitdr] : [uitdr])[GP-rij] []
[60] [cyclus]	<u>is</u>	MET[uitdr], [GP] := [uitdr], [uitdr] [programmadeel] ⁿ [nummer]HERHAAL
[70] [SUB-roep]	<u>is</u>	[SUB-naam]
[71] [SUB-beschrijving]	<u>is</u>	[nummer]SUB[SUB-naam] [] [opdracht] ⁿ [nummer]END []
[72] [SUB-naam]	<u>is</u>	[naam]
[80] [FUN-roep]	<u>is</u>	[FUN-NAAM][lijst aktuelen] <u>of</u> [FUN-NAAM] <u>of</u> [FUN-naam][lijst aktuelen]
[81] [FUN-NAAM]	<u>is</u>	[NAAM]
[82] [FUN-naam]	<u>is</u>	[naam]
[83] [lijst-aktuelen]	<u>is</u>	([aktuelen])
[84] [aktuelen]	<u>is</u>	[aktuele] <u>of</u> [aktuelen] ,[aktuele]
[85] [aktuele]	<u>is</u>	[uitdr] <u>of</u> [GP-rij]
[86] [FUN-beschrijving]	<u>is</u>	[nummer]FUN[FUN-naam][lijst formelen] [] [opdracht] ⁿ [nummer]FUN := [uitdr] [] <u>of</u> [nummer] []
[87] [lijst formelen]	<u>is</u>	([formelen])
[88] [formelen]	<u>is</u>	[formeel] <u>of</u> [formelen] ,[formeel]
[89] [formeel]	<u>is</u>	[GP] <u>of</u> RIJ[GP-rij]

INDEX

adres 6 10.2 24
afroonden 9
aktuele 10,3 22
binnencyclus 14
compiler 2 6 7
context regels 1 7 8 20 23
cyclus 14 20
descriptor 6 13 24
doeladres 11
EN 12
executie 3 9
formele 22 23
funktiedeel 4 23
FUN 19 22
FUN-beschrijving 4 7 8 12 14 22 23
FUN-roep 7 8 10.3 22 23 24
geheugenplaats, GP 6
getal 10.1
hoofdprogramma 4 7 8 20 23
index 10.2
index expressie 10.2
inhoud van GP 6
invoer 16
kommentaar 5.2
label 5.1 5.2 8 22 23
lokaal 22
naam 7 22 23
OF 12
ongereserveerd 13 21
ONWAAR 12
parameter 23
primitiva 10.5
prioriteit 10.4
produktie-regel VI
recursie 25 VI

RIJ,RIJSYM,RIJ-reserv. 3 6 7 13 14 23
spatie 5.3 VI
sprong 8 12
standaardfuncties 18 19
SUB 19
SUB-beschrijving 3 7 8 12 14 20 23
SUB-roep
subroutine 20
syntax 1 VI
tabel van symbolen 16
tekstuele volgorde 2 7 23
toekenning 11
uitdrukking 10
uitvoer 17
variabele 6 10.2
vertaalfase 2
WAAR 12
waarde 6 9 10.2 10.4

In deze index zijn de nummers paragraafnummers.
Als naar meerdere nummers verwezen wordt zijn
de belangrijkste onderstreept.

4 Wetenswaardigheden

4.1 LITERATUURBESPREKING

In het volgende worden in het kort enkele boeken besproken, die tot de redelijk eenvoudige leesstof gerekend moeten worden.

Tot en met 4.1.13 is Drs. C.A.Ch. Görts Uw redakteur.

4.1.1 'Computerwiskunde' f 3,50

van Prof.Dr.J.J. Seidel

uitgave Het Spectrum, Utrecht - Antwerpen
Aula boeken 407 - Nederlands - 155 blz.

Algemene inleiding over wiskundige hulpmiddelen die bij de computer van belang zijn. Het boek is ontstaan als heroriënteringscursus van de CMLW in 1967-1968.

Inhoud:

- 1) Algoritmen (Prof.Dr. A. v.d. Sluis)
- 2) Numerieke toepassingen (Prof.Dr.A.v.d.Sluis)
- 3) Non numerieke computertoepassingen (Prof.Dr. N.G. de Bruijn)
- 4) Over de structuur van rekenautomaten (Prof.Dr. E.W. Dijkstra).

4.1.2 'Assemblers and Loaders'

van D.W. Barron

uitgave MacDonald, Londen

Engels - 62 blz. - prijs f 14,25

Bij het programmeren in machinetaal schrijft de programmeur niet de bit-patronen op van de instructies, maar kan hij gebruik maken van lettercodes om het soort instructie aan te geven, van decimale getallen voor de adressen enz. Dit boek behandelt de vertaling van zulke machinetaal in de uiteindelijke bitpatronen. Enige kennis van de structuur van machinetalen wordt vereist.

4.1.3. 'Compiling Techniques'

van F.R.A. Hopgood

Engels - 115 blz. - prijs f 19,60

Compiling is het vertalen van een programma geschreven in een probleemgeoriënteerde taal (b.v. ALGOL) naar machinetaal. Dit boek geeft in een compacte, en daarom niet altijd gemakkelijke stijl een inleiding op dit terrein. Het boek veronderstelt bij de lezer enige kennis op het gebied van programmeertalen.

4.1.4 'Digitale automaten'

van W. Kämmerer

uitgave Akademie Verlag Verlin 1969

Duits - 482 blz. - prijs f 40,15

Een degelijk studieboek, bedoeld om een leek een basis te geven voor een werkkring op computergebied.

Inhoud:

- 1) Theorie (theoretische begrippen als berekenbaarheid, Turing machine, Boolese algebra)
- 2) Structuur (voorstelling van informatie in de computer, arithmetiek, structuur van computer, opdrachten)
- 3) Techniek (poorten, flip flops, kernen magneetbanden enz.)
- 4) Programmeren (programmeren in machinetaal; hogere programmeertalen en hun vertaling in machinetaal).

4.1.5 'Inleiding tot computer en automatisering'

van Prof.Dr. M. Euwe

uitgave Samson n.v., Alphen/Rijn, Brussel 1966

Nederlands - 126 blz. - prijs f 9,30

Algemene oriëntatie in computerwereld. Tevens bedoeld als leerboek bij het MEAO.

Inhoud:

- 1) Het nauwkeurig formuleren
- 2) Probleemanalyse
- 3) Schematechnieken (blokschema's)
- 4) De computer (opbouw van computer; rekenen in binair stelsel)
- 5) Het programma (beginnselen van programmeren in de machinetaal van de denkbeeldige machine SERA. Deze taal wordt veel gebruikt bij cursussen van het Studie Centrum voor Informatica).
- 6) Invoermiddelen (ponskaart, ponsband, magneetband).

4.1.6. 'List processing'

van J.M. Foster

uitgave MacDonald, Londen

Engels - 54 blz. prijs f 12,40

List processing is een programmeertechniek, waar-

INDEX

adres 6 10.2 24
 afronden 9
 aktuele 10,3 22
 binnencyclus 14
 compiler 2 6 7
 context regels 1 7 8 20 23
 cyclus 14 20
 descriptor 6 13 24
 doeladres 11
 EN 12
 executie 3 9
 formele 22 23
 funktiedeel 4 23
 FUN 19 22
 FUN-beschrijving 4 7 8 12 14 22 23
 FUN-roep 7 8 10.3 22 23 24
 geheugenplaats, GP 6
 getal 10.1
 hoofdprogramma 4 7 8 20 23
 index 10.2
 index expressie 10.2
 inhoud van GP 6
 invoer 16
 kommentaar 5.2
 label 5.1 5.2 8 22 23
 lokaal 22
 naam 7 22 23
 OF 12
 ongereserveerd 13 21
 ONWAAR 12
 parameter 23
 primitiva 10.5
 prioriteit 10.4
 produktie-regel VI
 recursie 25 VI

RIJ, RIJSYM, RIJ-reserv. 3 6 7 13 14 23
 spatie 5.3 VI
 sprong 8 12
 standaardfuncties 18 19
 SUB 19
 SUB-beschrijving 3 7 8 12 14 20 23
 SUB-roep
 subroutine 20
 syntax 1 VI
 tabel van symbolen 16
 tekstuele volgorde 2 7 23
 toekenning 11
 uitdrukking 10
 uitvoer 17
 variabele 6 10.2
 vertaalfase 2
 WAAR 12
 waarde 6 9 10.2 10.4

In deze index zijn de nummers paragraafnummers.
 Als naar meerdere nummers verwezen wordt zijn
 de belangrijkste onderstreept.

4 Wetenswaardigheden

4.1 LITERATUURBESPREKING

In het volgende worden in het kort enkele boeken besproken, die tot de redelijk eenvoudige leesstof gerekend moeten worden.

Tot en met 4.1.13 is Drs. C.A.Ch. Görts Uw redakteur.

4.1.1 'Computerwiskunde' f 3,50

van Prof.Dr.J.J. Seidel

uitgave Het Spectrum, Utrecht - Antwerpen
Aula boeken 407 - Nederlands - 155 blz.

Algemene inleiding over wiskundige hulpmiddelen die bij de computer van belang zijn. Het boek is ontstaan als heroriënteringscursus van de CMLW in 1967-1968.

Inhoud:

- 1) Algoritmen (Prof.Dr. A. v.d. Sluis)
- 2) Numerieke toepassingen (Prof.Dr.A.v.d.Sluis)
- 3) Non numerieke computertoepassingen (Prof.Dr. N.G. de Bruijn)
- 4) Over de structuur van rekenautomaten (Prof.Dr. E.W. Dijkstra).

4.1.2 'Assemblers and Loaders'

van D.W. Barron

uitgave MacDonald, Londen

Engels - 62 blz. - prijs f 14,25

Bij het programmeren in machinetaal schrijft de programmeur niet de bit-patronen op van de instructies, maar kan hij gebruik maken van lettercodes om het soort instructie aan te geven, van decimale getallen voor de adressen enz. Dit boek behandelt de vertaling van zulke machinetaal in de uiteindelijke bitpatronen. Enige kennis van de structuur van machinetalen wordt vereist.

4.1.3. 'Compiling Techniques'

van F.R.A. Hopgood

Engels - 115 blz. - prijs f 19,60

Compiling is het vertalen van een programma geschreven in een probleemgeoriënteerde taal (b.v. ALGOL) naar machinetaal. Dit boek geeft in een compacte, en daarom niet altijd gemakkelijke stijl een inleiding op dit terrein. Het boek veronderstelt bij de lezer enige kennis op het gebied van programmeertalen.

4.1.4 'Digitale automaten'

van W. Kämmerer

uitgave Akademie Verlag Berlin 1969

Duits - 482 blz. - prijs f 40,15

Een degelijk studieboek, bedoeld om een leek een basis te geven voor een werkkring op computergebied.

Inhoud:

- 1) Theorie (theoretische begrippen als berekenbaarheid, Turing machine, Boolese algebra)
- 2) Structuur (voorstelling van informatie in de computer, arithmetiek, structuur van computer, opdrachten)
- 3) Techniek (poorten, flip flops, kernen magneetbanden enz.)
- 4) Programmeren (programmeren in machinetaal; hogere programmeertalen en hun vertaling in machinetaal).

4.1.5 'Inleiding tot computer en automatisering'

van Prof.Dr. M. Euwe

uitgave Samson n.v., Alphen/Rijn, Brussel 1966

Nederlands - 126 blz. - prijs f 9,30

Algemene oriëntatie in computerwereld. Tevens bedoeld als leerboek bij het MEAO.

Inhoud:

- 1) Het nauwkeurig formuleren
- 2) Probleemanalyse
- 3) Schematechnieken (blokschema's)
- 4) De computer (opbouw van computer; rekenen in binair stelsel)
- 5) Het programma (beginnselen van programmeren in de machinetaal van de denkbeeldige machine SERA. Deze taal wordt veel gebruikt bij cursussen van het Studie Centrum voor Informatica).
- 6) Invoermiddelen (ponskaart, ponsband, magneetband).

4.1.6. 'List processing'

van J.M. Foster

uitgave MacDonald, Londen

Engels - 54 blz. - prijs f 12,40

List processing is een programmeertechniek, waar-

bij op objecten in het geheugen wordt geopereerd met inachtneming van de relaties, die er tussen de objecten bestaat. Voor deze techniek, die in niet-numerieke toepassingen veel gebruikt wordt, zijn speciale programmeertalen ontwikkeld. Ook verschillende algemene programmeertalen bieden faciliteiten op het gebied van list processing. Dit boekje veronderstelt enige basiskennis op het gebied van programmeren.

4.1.7. 'Primer of computing'

van Donald Leach en Bob Bowker

uitgave Holmes Mc.Dougall, Edinburg

Engels - 127 blz. - prijs onbekend

Hierin wordt iets verteld over de geschiedenis van de computer, hardware, programmeren in een eenvoudige machinecode, programmeertalen, toepassingen.

Er wordt niet diep op de zaken ingegaan. Misschien aardig voor de leerlingenbibliotheek (als de taal hiervoor geen bezwaar is).

4.1.8 'Inleiding tot IBM informatie verwerkende systemen'

Uitgave IBM

Nederlands - 107 blz. - prijs ± f 40,--

Deze handleiding is bedoeld als hulpmiddel bij het zich eigen maken van een basiskennis van computers. Besproken worden o.a. computertalstelsels, geheugeneenheden en hun werking, in- en uitvoereenheden, schrijven van programma's, programmeersystemen.

Uit dit uiteraard hier en daar op IBM apparatuur gericht boek zal men heel wat achtergrondinformatie kunnen halen. Het is voor serieuze belangstellenden, waaronder ook de experimenterende leraren worden gerekend, gratis verkrijgbaar bij IBM Nederland N.V., Afd. Publiciteit (t.a.v. N.A.M. v.d. Broek), Joh.Huizingalaan 257, Amsterdam.

4.1.9 'Elektronische Rekenmachines'

van P.L. Latour

uitgave Marka Boeken, Utrecht/Antwerpen

Nederlands - 218 blz. - prijs f 4,--

Een heel bruikbaar boek om wat achtergrondinformatie te verkrijgen. Behandelt o.a. de opbouw van de computer, machine-opdrachten, elektronische bouwstenen, constructie van geheugens,

in- en uitvoerorganen, algoritmische talen, analoge rekenmachines, kunstmatige intelligentie.

4.1.10 'Vertalen met de computer'

van Dr. F.G. Droste

uitgave Wolters Noordhoff N.V., Groningen - 1969

Nederlands - 169 blz. - prijs f 17,25

Een inleiding tot wat er in verschillende centra gedaan is aan machinale vertaling. Vakkennis op dit gebied noch op dat van computers is vereist.

4.1.11 'Hoe werkt een computer'

van R. Lohberg

f 15,75

uitgave Kluwer, Deventer/Antwerpen

Nederlands (uit Duits vertaald) - 235 blz.

Behandelt o.a. onderdelen van computer en hun werking, opdrachten, toepassingen. Gaat niet diep, maar bevat wel nuttige informatie. Maar de poging de zaak lichtverteerbaar te maken door een gezocht humoristische stijl van schrijven werkt wel storend.

4.1.12 'Computers and problem solving'

van T.E. Hull en D.D.F. Day

uitgave Addison Wesley, Canada - 1970

Engels - 276 blz. - prijs f 24,35

Het doel van dit boek is een leek de grondslagen van computergebruik bij te brengen. In het eerste deel wordt men vertrouwd gemaakt met programmeren; de gekozen programmeertaal is FORTRAN. Het tweede deel bevat een aantal toepassingen; deze zijn niet altijd volledig uitgewerkt; men kan dan desgewenst aan de hand van opgaven dieper op de zaak ingaan. De onderwerpen in deel II zijn: numerieke berekeningen, statistiek, simulatie, data processing, niet-numerieke berekeningen, toepassingen in computer science. In appendices komen o.a. ter sprake programmeren in assembleertaal en voorstelling van getallen. Bestudering van dit prettig geschreven boek, met name van deel II en de appendices, is ongetwijfeld erg nuttig, vooral als men bereid is ook de opgaven goed te bekijken.

4.1.13 'Inleiding tot de automatische informatieverwerking'

van A.B. Frielink

uitgave Agon Elsevier, Amsterdam/Brussel 1968
Nederlands - 118 blz. - prijs f 17,50

Een belangrijk gedeelte van dit boek wordt besteed aan de opbouw van de computer, de voorstelling van data in het tweetallig stelsel, het rekenorgaan en het programmeren. Bij het programmeren wordt de taal SERA gebruikt, een machinetaal voor een fictieve computer, ontworpen ten behoeve van de programmeursopleiding. Daarnaast komen onderwerpen aan de orde als voorbereiding van een computertoepassing, typen van computers, invoerproblematiek, organisatie rekencentrum, historisch overzicht, theoretische grondslagen. Waarschijnlijk omdat de auteur in een kleine omvang veel heeft willen aansnijden, zijn de meeste onderwerpen wel erg beknopt behandeld. Het boekje kan van nut zijn als men het vak eens van een andere zijde (administratief gebruik) belicht wil zien.

4.1.14 'Eenvoudige Computerkunde'

van A. Goudvis

uitgave J. Rijkhoek, Den Haag
Nederlands - 110 blz. - prijs f 5,--.

De schrijver geeft 110 pastelkleurige bladzijden informatie over de computer en zijn toepassingen. Na een kort historisch overzicht en enige instructie over het tweetallig stelsel volgt een opsomming van functie en werking van veel toestellen, die men in de omgeving van een computer kan aantreffen. Daarna volgt de organisatie van de rekenautomaat en iets over de wijze, waarop de machine door de mens wordt geïnstrueerd. De schrijver kan in een zo kort bestek over de onderwerpen niet volledig zijn, maar hij doet - gezien het grote aantal ingevoerde begrippen wel een poging in die richting. Daarna spitst het betoog zich toe op het aanbrenge van een juist begrip van de toepasbaarheid van computers in het bedrijfsleven. Het boek is volgens de ondertitel bedoeld als leerboek in computerkunde voor de hoogste klassen van de middelbare school en voor de H.T.S. Als leerboek bevat het te veel gegevens in een kort bestek en te weinig (namelijk helemaal geen) oefenmateriaal om instructief te zijn. Het leervak 'computerkunde' zal meer moeten bevatten dan de door de schrijver gegeven feitenkennis van computers. Het zal in

de eerste plaats de - ook buiten de automatisering belangrijke - vaardigheden moeten aanbrenge op operationeel (duidelijke, ondubbelzinnige instructies geven), algoritmisch (herkennen van routinematigheden) en organisatorisch gebied. Aan deze facetten is de schrijver helaas voorbijgegaan. Dit neemt niet weg, dat het boek aanbevolen kan worden als leesboek voor de leerlingenbibliotheek.

4.1.15 'Elementaire computerkunde voor mavo en havo' deel I en II

Drs. W. van den Camp, H.B. Emanuels,
J.B. Lubke, P.Th.J. Ploeger
Uitg. Meulenhoff, Amsterdam.
f. 7.90 en f. 13.90

De inhoud van beide delen is in grote lijnen als volgt:

Het eerste deel begint met een elementaire uiteenzetting over wat een computer is en hoe deze werkt; vervolgens wordt het begrip algoritme besproken, waarna verder veel aandacht wordt besteed aan programmastroomschema's enerzijds en (een subset van) SERA-69 anderzijds. In het tweede deel wordt begonnen met een behandeling van ALGOL. Deze behandeling draagt het karakter van een 'eerste' ronde cursus, waarbij de procedures in het geheel niet ter sprake komen. Verder worden verschillende voorbeelden uitgebreid besproken. Het tweede deel besluit met twee hoofdstukjes over de sociale aspecten van de automatisering. Vooraan in het tweede deel doen de schrijvers een voorstel tot behandeling van de geboden leerstof in de vorm van een gemeenschappelijk basisstuk (met onderwerpen uit zowel het eerste als tweede deel), gevolgd door een verdere uitbouw in twee richtingen waarvan de een meer wiskundig en de ander meer administratief is gericht. Bij nadere beschouwing blijken de twee richtingen maar weinig van elkaar te verschillen. De splitsing doet dan ook wat geforceerd aan. Gezien de groepen, waarvoor de 'Elementaire computerkunde' is bedoeld lijkt deze splitsing ook niet erg zinvol. Keuze van onderwerpen is wel zinvol en eventueel zelfs nodig, als deze dan maar geschiedt op basis van niveau en belangstelling van de groep in kwestie.

Afgezien van genoemd (detail-) bezwaar lijkt mij het voorstel alleszins de moeite van het overwegen waard. De schrijvers hebben in ieder geval de mogelijkheid er toe geschapen, door de lesstof in niet al te grote, duidelijk afgeronde,

eenheden te presenteren. Gelukkig wordt in het voorstel het SERA-gedeelte als een op zichzelf staande eenheid beschouwd. Het is namelijk zeer de vraag of SERA (zij het tot BABY of JR gereduceerd) in een cursus voor het secundaire onderwijs thuishoort.

Enig inzicht betreffende de werking in de computer, zo al nodig, kan m.i. op een meer eenvoudige manier worden bijgebracht.

Als opvallend goede onderdelen kunnen worden genoemd:

- a. de behandeling van programmastroomschema's. Zowel in lesstof als vraagstukken is sprake van een didactisch goed doordachte opbouw. De schrijvers hebben blijkbaar bij het uittesten een schat van ervaring opgedaan. Verwacht kan worden dat leerlingen (en docenten!) aan de duidelijke, goed gedoseerde en rijkelijk genuanceerde weergave van deze ervaring een sterke motivatie tot het verwerven van eigen inzicht zullen ontleenen. Dit laatste geldt ook en zeker niet in mindere mate voor
- b. de meer uitgebreide voorbeelden in het tweede deel. Zowel de afwisseling van onderwerpen als de manier van behandeling maken dat de kwalificatie *boeiend* hier terecht van toepassing is. Vooral de manier waarop hierbij gebruik wordt gemaakt van stroomschema's, getuigt van de grote zorg waarmee de schrijvers deze voorbeelden en bijbehorende vraagstukken hebben samengesteld.

In het voorwoord van het tweede deel wordt gesteld dat beide delen een experimenteel karakter hebben. Mogelijk is het daaraan te wijten dat, onverlet de grote waardering voor het geheel en de meeste onderdelen, er toch ook enkele onderdelen zijn, die van de nodige kritische kanttekeningen moeten worden voorzien. De schrijvers verwachten (voorwoord deel I) dat de leerlingen grote delen van de stof zelfstandig kunnen doorwerken. Maar dan is het wel nodig dat, zeker waar het gaat om een eerste kennismaking, verwarring scheppende onduidelijkheden en niet waar te maken uitspraken worden vermeden. Het eerste heeft vooral betrekking op de hoofdstukken 3 (Wat is een computer) en 4 (Hoe werkt een computer) van het eerste deel, het laatste op de twee laatste hoofdstukken van het tweede deel; die handelen over de sociale aspecten en de automatisering. De gesignaleerde tekortkomingen doet de overtuigend goede kwaliteit van de 'Elementaire Computerkunde' nauwelijks geweld aan. De boekjes kunnen dan ook van harte worden aanbevolen, met name voor de in de titel genoemde schoolsoorten: Lavo, Havo.

F. Remmen

4.1.16 'Inleiding in de informatica'

Drs. F. Remmen

Uitg. H.E. Stenfert Kroese N.V. Leiden
f. 10.--

Het hier besproken boek is ontstaan uit een syllabus voor eerstejaarsstudenten aan de H.E.A.O.-Arnhem.

Zoals het voorwoord aangeeft is het bedoeld voor het bijbrengen van de nodige basisbegrippen. Bijzonder prettig is dat naast de nederlandse benamingen van deze begrippen ook de engelse equivalenten worden gegeven.

Hoewel dit niet geheel in het voorwoord van het boek tot uitdrukking komt, is de behandelde stof in eerste instantie bestemd voor eerstejaars H.E.A.O.-studenten; daarnaast is het boek echter goed bruikbaar om basisbegrippen bij te brengen aan degenen die een eerste (theoretische) kennismaking wensen met de automatisering.

Het zelfstandig doorwerken van het boek acht ik niet goed mogelijk, ook vanwege het feit dat de grote verzameling opgaven, waarvan de antwoorden niet in het boek zijn opgenomen, een essentieel bestanddeel van het boek vormen.

Ik zou mijn waardering willen uitspreken over de korte en zakelijke opzet van het boek, geschreven in een prettige stijl. Het boek telt, inclusief register en literatuurlijst, 94 pagina's. Er wordt in tegenstelling tot vele andere publicaties die beogen een inleiding te zijn, in deze 94 bladzijden een schat van gegevens overgedragen. Dat de compactheid somtijds bezwaren oplevert, zal echter ook duidelijk zijn.

Naar mijn mening komt het onderwerp systeemanalyse en -ontwerp nauwelijks aan de orde. Hoewel ik mij bewust ben dat het op het niveau van degenen voor wie dit boek bestemd is, een bijzonder moeilijk te behandelen onderwerp is, vindt ik het een zo essentieel onderdeel van de informatica, dat het in een inleiding beslist niet mag ontbreken. Bovendien kan het dienen als een 'kader' waarin een groot aantal onderwerpen een plaats kunnen krijgen. Nu zijn onderwerpen als programmeren, systeemstroomschema's, e.d. wat moeilijk plaatsbaar. Bovendien krijgt de hardware een te zwaar accent. In het boek worden hier dan ook ruim 40 pagina's aan gewijd.

Aan de onderwerpen systeemstroomschema's en programmastroomschema's 14 pagina's.

Opvallend zijn de hoofdstukken Bestandsbeheer, als inleiding op de externe geheugens en het hoofdstuk Beslissingstabellen.

Programmeren komt slechts in een hoofdstukje van

6 pagina's aan de orde. De (goede) reden die voor deze beknoptheid wordt gegeven, is dat de schrijver meent dat de keuze van de te behandelen programmeertaal momenteel nog te veel afhankelijk is van plaatselijke omstandigheden. Het leek de schrijver daarom niet zinvol in deze algemeen bedoelde inleiding een keuze te maken die voor sommigen (nog) niet bruikbaar zou zijn. Hij merkt daarbij nog op dat er op het gebied van de programmeertalen al diverse goede handleidingen bestaan. Het zou overigens prettig zijn geweest als de schrijver in zijn literatuurlijst een aantal handleidingen op het gebied van de programmeertalen had opgenomen.

Een ander onderwerp dat ik graag iets uitgebreider behandeld had willen zien, is het bedrijfs-systeem van de computer. Hier en daar komt het wel aan de orde, maar gezien de belangrijke plaats die het bedrijfssysteem van de computer zo langzamerhand inneemt, wat te summier. Samenvattend zou ik willen stellen dat het gedeelte van het boek dat de apparatuur behandelt, het sterkste deel van het boek is. Bovendien is een leuk, en toch kort, hoofdstukje over beslissings-tabellen opgenomen.

P.Th.J. Ploeger

4.2 TERMINALGEBRUIK BIJ DE COMPUTERKUNDE

Wat is een terminal?

Het maakt de indruk van een schrijfmachine op een metalen tafeltje, meestal als één geheel uitgevoerd. Van de meeste merken is het schrijfmachine gedeelte van dezelfde soort als een elektrische schrijfmachine. Men ziet de schrijfkop langs het papier bewegen en symbolen op het papier slaan (Compk 1, foto op blz. 89).

Elk door de gebruiker aangeslagen symbool verschijnt also op het papier, maar wordt tevens omgezet in een soort Morse-code. En deze code wordt via een draad, meestal een telefoonlijn, verzonden naar een rekencentrum. In het rekencentrum worden de binnenkomende signalen opgevangen, door een apparaat dat in wezen een kleine computer is. Hierin worden de signalen onderzocht op defecten, die door een storing op de lijn ontstaan kunnen zijn. De code heeft namelijk een ingebouwde 'check' tegen verminkingen. Iets soortgelijks heeft bijvoorbeeld ook de ponsbandcode van Compk 1 blz. 37, waarbij het aantal gaatjes van een symbool (boven elkaar) altijd oneven is. Het ontbreken van een gaatje wordt ogenblikkelijk ontdekt. Maar het ontbreken van twee gaatjes wordt dan toch maar niet ontdekt, zegt U nu en gelijk hebt U. Als de kans op het ontbreken van een gaatje 1 op 1000 is, is de kans op 2 verdwenen gaatjes in eenzelfde teken 1 op 1000^2 , en dat risico neemt men. Overigens bestaan er codes, die zo zijn opgebouwd, dat zij lichte verminkingen overleven. Bij aankomst kan het gezonde signaal er weer uit worden afgeleid. Maar wij zijn wat afgedwaald naar de ziekenhuissfeer. Terug naar het rekencentrum.

Het computertje is niet alleen verbonden met onze terminal, maar ook met andere eindstations, die gelijktijdig (time sharing) zich mogen aanmelden. Maar het apparaat is ook geprogrammeerd om de verschillende stations uit elkaar te houden.

Ook het verzenden van informatie in omgekeerde richting is mogelijk, waarbij dan echter geen fout-detectie bij het eindstation mogelijk is. Een kraakje op de lijn geeft al gauw een onleesbare regel.

Hoe gaat de bediening in zijn werk?

Men belt een bepaald nummer en hoort aan een bepaald toontje dat men aangesloten is op het computersysteem. Men legt de hoorn naast het telefoontoestel (of bij sommige portable terminals in een daarvoor bestemde opening) en schakelt de terminal in. Het eerste wat men in moet tikken is een geheim deelnemersnummer of iets dergelijks. Dit om te voorkomen dat iedereen die een terminal in elkaar knutselt naar willekeur van allerlei computers gratis gebruik zou maken. Na deze legitimatie en mogelijke andere formaliteiten kunt U beginnen met het intikken van programma's. Er zijn systemen waarbij symbool voor symbool wordt doorgezonden, andere waarbij eerst een hele regel wordt opgespaard in een geheugentje en daarna in zijn geheel verzonden. Zoals gezegd: bij verminkte overkomst krijgt U onmiddellijk reactie op Uw toestel door middel van een foutmelding. U kunt de regel opnieuw tikken.

Als U Uw programma hebt voltooid en U laat een startsein volgen, dan wordt het programma aangeboden aan de computer. De uitkomsten verschijnen onmiddellijk na berekening op Uw terminal, tenzij fouten in het programma dat verhinderen, maar dan ontvangt U de foutmelding. Onmiddellijk herstel van fouten is dus mogelijk.

Voorbeeld van een programma

Hieronder ziet U een programma in de programmeertaal FORTRAN voor de berekening van een kwadraten tabel. READY geeft aan dat het systeem klaar staat

om de regel te ontvangen, die er achter wordt ingetikt. Dit is dus informatie door ons weggezonden. De terug ontvangen regels zijn aangegeven met YIELD. Het programma zou in ECOL luiden:

```

START
k := 2
i := 2
5 j := i * i
  SPATIE(2)
  SCHRIJF(2,0) := k
  NR
  SCHRIJF(3,0) := i
  SPATIE(5)
  SCHRIJF(6,0) := j
  ALS i=50 DAN 20
  i := i + 1
  NAAR 5
20 KLAAR

```

U ziet dat men in FORTRAN evenals in ECOL in de voorwaardelijke sprongopdracht het ANDERS mag weglaten. Voor het geval dat $i \neq 50$ gaat het programma bij de volgende regel verder. In regel 105 moet GOF"TO gelezen worden als GOTO. De leerling van de Prot.Chr. Mavo in Amsterdam-Noord, die abusievelijk F schreef, heeft dit hersteld. Verder ziet U dat de SCHRIJF-opdracht veel korter geformuleerd is dan in ECOL; er staat zoiets als:

103. SCHRIJF (op terminal 6, volgens regel 15)
k, i en j
104. 15 FORMAAT (2 spaties, geheel getal van 2 cijfers, nieuwe regel, geheel getal van 3 cijfers, 5 spaties, geheel getal van 6 cijfers)

LINE	STATUS	CONVERSATIONAL FORTRAN
100.	READY	K=2
101.	READY	I=2
102.	READY 5	J=I*I
103.	READY	WRITE(6,15)K,I,J
104.	READY 15	FORMAT (2X,I2,/,I3,5X,I6)
105.	READY	IF (I.EQ.50) GOF"TO 20
106.	READY	I=I+1
107.	READY	GOTO 5
108.	READY 20	STOP
109.	READY	END
110.	READY #BEGIN	
103.	YIELD 2	
103.	YIELD 2	4
103.	YIELD 2	
103.	YIELD 3	9
103.	YIELD 2	
103.	YIELD 4	16
103.	YIELD 2	
103.	YIELD 5	25
103.	YIELD 2	
103.	YIELD 6	36

.....

Gebruik van de terminal

Het grote voordeel van een terminal heeft U uit het voorgaande kunnen afleiden. De leerling heeft 'lijfelijk kontakt' met apparatuur en ziet onmiddelijk resultaten.

Als nadelen zijn aan te merken:

- het toetsenbord en de verdere bediening van het toestel kunnen voor leerlingen zonder ervaring met een typemachine een hoge drempel zijn
- één terminal per 30 leerlingen blijkt onvoldoende
- de bediening van de terminal moet in veel gevallen buiten het lesuur computerkunde plaatsvinden (per lesuur kunnen immers hooguit 4 leerlingen hun programma's laten verwerken)
- het zijn vaak alleen de leerlingen met grote belangstelling voor computers, die de terminal de meeste tijd bezetten. De meerderheid van onze leerlingen, die we toch een indruk van computers willen geven en waarop ons onderwijs zich juist richt, raken in dat geval in de verdrukking.
- een schatting van de totale kosten van f 600,- per maand is aan de lage kant.

Het is zaak de genoemde bezwaren zoveel mogelijk op te heffen, want de terminal zal in de wat verdere toekomst beslist zijn weg vinden naar onze scholen.

5 Achtergrondinformatie

5.1 COMPUTERHARDWARE

Hieronder vindt U de syllabus van het college, zoals dit op de heroriënteringscursus voor leraren gegeven werd door Prof. Dr. A. van der Sluis.

1. Geheugen

Het geheugen van een computer bestaat meestal uit een werkgeheugen en een achtergrondgeheugen. In kantoortermen: het werkgeheugen is zoiets als de kaartenbak die de boekhouder voor zich heeft staan, het achtergrondgeheugen is zoiets als het archief op zolder. Het werkgeheugen is niet zo groot, maar zeer snel toegankelijk, het achtergrondgeheugen is zeer groot maar niet zo snel toegankelijk.

1.1. Werkgeheugen

Het werkgeheugen bestaat doorgaans uit een groot aantal (enkele duizenden tot ettelijke miljoenen) ferrietringetjes. Zo'n ringetje kan linksom of rechtsom gemagnetiseerd zijn en gedraagt zich dus als een binaire variabele. We zullen de ene magnetisatie richting 0 en de andere 1 noemen. We noemen zo'n binaire variabele ook een bit (binary digit).

Deze bits worden in groepjes ingedeeld. Een zo'n groepje heet een woord. We zullen in deze syllabus de bits van één woord steeds nummeren 0, 1, 2, ..., en als we zo'n woord op papier schrijven zullen we steeds bit 0 rechts schrijven, bit 1 links daarvan etc. We zullen ook bit 0 het minst significante bit noemen en het meest linkse bit het meest significante.

Het aantal bits waaruit een woord bestaat varieert van het ene machinetype tot het andere en heet de woordlengte. Voor komende woordlengtes zijn 12, 16, 18, 24, 27, 32, 36, 48, 64 en misschien nog wel andere. Hiernaast kwamen vooral vroeger computers voor met variabele woordlengte, d.w.z. dat de programmeur zelf de indeling van al de geheugenbits in woorden kon vaststellen.

De woorden worden genummerd 0, 1, 2, ... Het nummer van zo'n woord noemt men zijn adres.

Gewoonlijk heeft een woord nog een extra bit, het zgn. pariteitsbit. Bij het opbergen van informatie in het geheugen wordt dit bit steeds 1 gemaakt als de op te bergen informatie een even aantal enen bevat, anders 0 (of steeds andersom). Dit gebeurt automatisch. Bij het uitlezen van informatie uit het geheugen controleert de computer of het totaal aantal enen de juiste pariteit heeft en slaat alarm als dit niet zo is.

Zo'n ringetjes geheugen is heel snel toegankelijk, des te sneller naarmate de ringetjes kleiner zijn. Men is tegenwoordig al onder de microseconde accestijd en bij $\pm 0,5$ mm buitendiameter van de ringetjes.

1.2. Registers

Naast het ringetjes geheugen heeft vrijwel elke computer nog een aantal uit flip-flops bestaande geheugenplaatsen welke men gewoonlijk registers noemt. Deze zijn nog sneller toegankelijk. Men meet de accestijd in nanoseconden.

1.3. Achtergrondgeheugen. Zie hiervoor 9.

2. Getalrepresentatie

Wat de rij nullen en enen, die de inhoud van een geheugenwoord vormen, voorstelt hangt geheel af van de opdrachten die op zo'n geheugenwoord opereren. Maar zelfs als een geheugenwoord alleen maar een getal hoeft voor te stellen gebeurt dit nog op allerlei verschillende manieren (afhankelijk van het type computer); de meeste computers hebben zelf nog weer verschillende manieren om getallen voor te stellen.

2.1. Gehele getallen

Er zijn twee standaard manieren om gehele getallen voor te stellen:

(a) 2-complement representatie: het woord $p_n, p_{n-1}, \dots, p_0$ stelt het getal $\sum_{i=0}^{n-1} p_i 2^i$ voor als $p_n=0$ en het getal $(\sum_{i=0}^{n-1} p_i 2^i) - 2^n$ als $p_n=1$.

Voorbeeld: $n=4$ $00101=5$ $10101=-11$.

We zullen ons in deze syllabus voornamelijk aan deze representatie houden, alhoewel in computers de volgende representatie gebruikelijker is:

(b) 1-complement representatie: het woord $p_n, p_{n-1}, \dots, p_0$ stelt het getal $\sum_{i=0}^{n-1} p_i 2^i$ voor als $p_n=0$ en het getal $-(\sum_{i=0}^{n-1} p_i 2^i)$ als $p_n=1$. Dit komt er in feite op neer dat als $p_n=1$ alle overige bits "omklappen": 0 wordt 1 en 1 wordt 0.

Voorbeeld: $n=4$ $00101=5$ $10101=-10$.

Voor $p_n=0$ stemmen (a) en (b) overeen, voor $p_n=1$ geldt (a)=(b)-1. Bij (b) zijn er twee representanten van het getal 0: allemaal nullen en allemaal enen.

Bij (b) heeft men geen speciaal aftrekmechanisme nodig: als men van de aftrekker alle enen in nullen verandert en andersom mag men optellen. Een opteller voor (b) is evenwel iets ingewikkelder dan een voor (a) zoals we nu zien.

Voorbeeld: $n=4$, representatie (a), tel op 5 en -3, dus 00101 en 11101 . De optelling wordt eenvoudig tot stand gebracht door de operanden gewoon als positieve binaire getallen te beschouwen en ze op te tellen. In feite ontstaat dan 100010 , maar door voor de som maar weer 5 bits ter beschikking te stellen ontstaat de meest linkse 1 helemaal niet en krijgt men het juiste antwoord $00010=2$. Analooft $-5+3:11011+00011=11110=-2$.

Voorbeeld: $n=4$, representatie (b), tel op 5 en -3, dus 00101 en 11100 . Gewoon optellen als zoëven geeft 00001 en dat is 1 te weinig. Evenwel geeft $-5+3$, dus $11010+00011=11101=-2$ het goede antwoord. Het geheim zit erin dat als bij het "gewoon optellen" een 6e bit 1 zou ontstaan, men op de meest rechtse plaats van het resultaat 1 moet bijtellen (end around carry).

Als het resultaat van optelling of aftrekking in absolute waarde te groot is ontstaan er moeilijkheden.

Voorbeeld: $n=4$, representatie (a) of (b), tel op 12 en 14, dus 01100 en 01110 . Dat kan helemaal niet, want 26 is niet voorstelbaar. De uitkomst 11010 wordt als negatief getal geïnterpreteerd. Men zegt dat er overflow optreedt.

Er is analogie met de kilometerteller van een auto. Als de auto nieuw is en op 00000 staat en men rijdt 5 km achteruit dan staat hij op 99995 en dat zal men wel willen identificeren met -5. Maar als men dat doet kan men het getal 99995 niet meer representeren.

Hoe groter de woordlengte, hoe groter getallen men nog kan representeren, maar een grens blijft er. In vele machines is er een soort alarm als er overflow optreedt.

2.2. Niet gehele getallen

Men wil natuurlijk ook met niet gehele getallen kunnen rekenen. Daartoe beschikken vele computers over een drijvende komma voorstelling van getallen. Een onrealistisch voorbeeld hiervan: woordlengte 9 bits. De bits 0 t/m 4 stellen een geheel getal voor (de mantisse), de bits 5 t/m 8 stellen de exponent voor van de macht van 2 waarmee de mantisse moet worden vermenigvuldigd.

Dan stelt $0000\ 0111$ gewoon 15 voor, maar $0001\ 0111$ stelt $2 \times 15 = 30$ voor, $0101\ 0101 = 2^3 \times 11 = 352$, $1101\ 0101 = 2^{-3} \times 11$, $1101\ 1101 = 2^{-3} \times (-5)$ wanneer men mantisse en exponent steeds volgens 2.1(a) interpreteert.

Dit voorbeeld is zeer onrealistisch door het geringe aantal bits dat voor mantisse en exponent beschikbaar is. Realistischer is: woordlengte 48 bits; mantisse 0 t/m 37, exponent 38 t/m 47. Dan ligt de exponent in tussen -512 en +511, zodat de 2 macht waarmee de mantisse moet worden vermenigvuldigd in

ligt tussen circa 10^{-153} en 10^{153} . Dit betekent dat men zeer grote en zeer kleine getallen nog met zeer goede precisie kan voorstellen.

Merk nog op dat ook gehele getallen uitstekend in drijvende kommagedaante zijn voor te stellen.

3. Waarom verschillende representaties

Dat men in een en dezelfde computer gehele getallen zowel in vaste komma (d.w.z. volgens 2.1) als in drijvende komma (d.w.z. volgens 2.2) voorstelling kan weergeven is een kwestie van tijdeconomie. Het ligt immers voor de hand dat de extra manipulaties met de exponent extra tijd kosten, vooral bij het optellen: de computer moet dan immers eerst de mantissen ten opzichte van elkaar verschuiven over een aantal posities dat met het exponentenverschil overeenkomt. Met name bij vele simpele telopdrachten (bijv. de teller in een cyclus) werkt men met vaste komma getallen veel sneller.

Hoe weet de computer nu of een getal dat hij uit het geheugen haalt in vaste of in drijvende komma voorstelling staat? Dat hangt af van de opdracht waarvoor dit getal uit het geheugen wordt gehaald; er zijn opdrachten die met drijvende komma getallen werken en opdrachten die met vaste komma getallen werken (zie 4.2).

Hoe kan men nu in een computer met woordlengte 12 of 16 (of zelfs 32) bits enigmatische redelijke getallen voorstellen. (2^{16} is immers slechts ≈ 64000). In zo'n computer gebruikt men meerdere woorden (gewoonlijk in opeenvolgende adressen) om één getal voor te stellen en zo krijgt men toch voldoende veel bits voor één getal beschikbaar.

Sommige computers hebben ook opdrachten die dergelijke meer-voudige lengte getallen "in een klap" uit het geheugen halen en verwerken, bij andere moet men dit geheel uitprogrammeren.

Bij het werken met ECOL laten we de computer uitsluitend met drijvende komma getallen werken. Zou men namelijk van de grotere efficiëncy van het werken met gehele getallen hebben willen profiteren, dan had men de leerling moeten verplichten aan het begin van zijn programma te verklaren (declareren) welke van zijn variabelen alleen gehele waarden hoeven te kunnen aannemen en welke niet (zie bijv. ALGOL). Dit hebben we hem willen besparen.

2.4. Decimale getallen

Sommige computers kunnen ook met decimale getallen werken en onder omstandigheden spreekt men ook van decimale computers. Dit betekent eenvoudig, dat men in de meest rechtse 4 bits (16 mogelijkheden!) de meest rechtse decimaal onderbrengt, in de volgende 4 bits de tweede decimaal etc.

Het nut hiervan is, dat met name bij administratief gebruik een computer met een getal vaak niet veel meer doet dan het aflezen en afdrukken en dan is het wel jammer dit getal, dat immers op een ponskaart decimaal wordt voorgesteld eerst naar de binaire vorm te converteren en dan weer terug. Het rekenen met decimale getallen gaat echter wel langzamer.

3. Andere soorten informatie

In het geheugen van een computer wil men niet alleen getallen kunnen opbergen, maar ook bijv. tekst en de opdrachten van het programma. Ook deze informatie moet men dus voorstellen door rijtjes bits. Ook deze voorstelling verschilt weer van machine tot machine. Een heel gebruikelijke situatie is dat men letters, cijfers, tekens door rijtjes van 8 bits voorstelt (zodat men dus 256 verschillende "karakters" kan voorstellen). Bij een woordlengte van 32 bits kan men dan 4 karakters in een woord onderbrengen. Zo'n woord dat met 4 karakters gevuld is is weer ononderscheidbaar van een woord met een getal erin. Het hangt weer af van de opdracht, die op zo'n woord opereert, hoe de informatie geïnterpreteerd wordt.

Omdat er zo vaak met karakters gemanipuleerd wordt is men er toe overgegaan machines te bouwen die niet alleen in woorden van (zeg) 32 bits adresseerbaar zijn, maar bovendien in "woorden" van 8 bits, zgn. bytes.

Zo zal het woord met adres 328 de bytes met adressen 1312 (=4x328), 1313, 1314 en 1315 bevatten. Aldus is men in staat

om met een opdracht 1 karakter uit een woord te halen. Zo kan de RIJSYM in ECOL door een rij bytes worden voorgesteld.

Maar ook kan een woord uit "losse" bits bestaan.

Een andere mogelijkheid is dat men in een programma met zeer vele kleine getalletjes, bijv. ≤ 31 , moet werken. Hiervan kan men er 6 achter elkaar in een woord van 32 bits stoppen.

4. Opdrachten van een computer

Het opdrachten-arsenaal van een computer is enerzijds veel beperkter en anderzijds veel uitgebreider dan de opdrachten die we in ECOL tegenkwamen.

De meeste computers hebben geen opdrachten van het type $a := b + c$, waarin a, b en c willekeurige geheugenplaatsen zijn, om nog maar te zwijgen van opdrachten als $a := (b+c) \times (d+e)$ of $a := \sin(x)$ of $a(i) := b(3 \times j + k) + c(r)$.

Wanneer we nu een schets geven van opdrachttypen die computers wel kennen, bedenke men dat de opdrachtersets van computers zeer verschillend zijn en er waarschijnlijk geen computer is die alle volgende opdrachten heeft, terwijl wij anderzijds niet volledig zijn.

In het volgende zullen we het geheugen beschouwen als een rij $M(0:n)$, n een of ander groot getal, zodat bijv. $M(8)$ de geheugenplaats met adres 8 is.

De registers (zie 1.2) zullen we aanduiden met $R_1, \dots, R_m$, waarin m gewoonlijk een vrij laag getal is, bijv. 7 of 16.

4.1. Transportopdrachten

Een computer kan gewoonlijk alleen maar informatie heen en weer transporteren tussen een geheugenplaats en een register, niet rechtstreeks van geheugenplaats naar geheugenplaats.

Dergelijke opdrachten hebben het type

- (1) $R_i := M(j + R_k)$ (2) $R_i := -M(j + R_k)$
(3) $M(j + R_k) := R_i$ (4) $M(j + R_k) := -R_i$

Hierin zijn i, j en k getallen. Dezelfde opdrachten, maar dan zonder de R_k zijn ook toegestaan. Om dit in het volgende niet telkens expliciet te hoeven zeggen voeren we een fictief register R_0 in, dat altijd het getal 0 bevat.

Kennelijk kan men dan deze opdrachten kenmerken door een viertal getallen, namelijk opdrachttype (we zullen dit getal verder f noemen) en de getallen i, k, j. Door voor elk van deze getallen een vast aantal bits beschikbaar te stellen wordt de hele opdracht eenduidig door een rij bits voorgesteld. Zo zouden deze opdrachten bijv. het volgende formaat kunnen hebben

f				i				k				j			
31	24	23	21	20	18	17									0

met 8 bits (dus 256 mogelijkheden) voor het opdrachttype f, 3 voor i en k en 18 voor j.

Gesteld dat de opdracht (1) de f-bits 00101001 heeft, zal het patroon

00101001 101 011 000000100010001101
f i k j

de opdracht

$$R_i := M(2189 + R_j)$$

voorstellen.

Het is nu ook duidelijk dat er nog weer verschillende opdrachten van bijv. gedaante (1) kunnen zijn:

- een die slechts 8 bits uit het aangeroepen woord ophaalt (omdat men bijv. met karakters wil werken en deze bits juist een karakter kenmerken), welke 8 bits nog weer op een aangegeven plaats in dit woord staan;
- een die een heel woord ophaalt;
- een die twee opeenvolgende woorden ophaalt (en deze ook in twee opeenvolgende registers deponaert).

Deze opdrachten onderscheiden zich dan namelijk in de eerste 8 bits.

De mogelijkheid geheugenplaatsen aan te wijzen door $j + R_k$ dient voor de rij organisatie. Zij a een rij zodat $a(0)$ op adres 80 staat $a(1)$ op 81 etc. en zij b een rij zodat $b(0)$

op adres 216 staat, b(1) op 217 etc. Laat de variabele p op geheugenplaats 12 staan en q op 27. Dan wordt een simpele ECOL opdracht als $a(p) := b(q)$ voorgesteld door de machine opdrachten $R_1 := M(27); R_2 := M(216+R_1); R_1 := M(12); M(80+R_1) := R_2$.

4.2. Optel- en aftrekopdrachten

Deze plegen van het type $a := b \pm c$ te zijn, waarbij b en/of c een register is en waarbij a hetzelfde is als b of c. Geheugenadressen zijn weer als onder 4.1: getal + R_K .

Dus bijv.:

$R_1 := R_1 - M(j+R_K)$

$M(j+R_K) := M(j+R_K) - R_1$

Er zijn weer verschillende opdrachten van dezelfde gedaante al naar gelang de betrokken bitrijen als vaste- of als drijvende komma getal moeten worden geïnterpreteerd of misschien zelfs de informatie in twee opeenvolgende geheugenplaatsen of registers als een geheel moet worden beschouwd.

Geheel getal optel- en aftrekopdrachten duren nauwelijks langer dan de geheugenaccessijds, in heel snelle computers zelfs aanmerkelijk korter. Drijvende komma optel- en aftrekopdrachten kunnen wel duidelijk langer duren: 30 μ sec. in een middelsnelle computer, 75 nanoseconde in de allersnelste.

4.3. Vermenigvuldig- en deelopdrachten

Voor de drijvende komma opdrachten geldt veelszins hetzelfde als bij 4.2, zij het dat van de royale verzameling opdrachten die aldus in het vooruitzicht wordt gesteld veelal slechts een paar gerealiseerd zijn, bijv. alleen

$R_1 := R_1 \times M(j+R_K)$

$R_1 := R_1 / M(j+R_K)$

en het verkregen resultaat is afgerond.

Bij vaste komma vermenigvuldigen heeft men meestal een opdracht als

$R_1 R_2 := R_1 \times M(j+R_K)$

waarin $R_1 R_2$ aanduidt dat de inhouden van R_1 en R_2 achter elkaar geschreven moeten worden gedacht.

Bij vaste komma delen heeft men veelal een opdracht als

$R_1 := R_1 R_2 / M(j+R_K), R_1 := \text{rest}$

zodat men met één opdracht quotient en rest ter beschikking krijgt.

Vermenigvuldigopdrachten variëren van 30 μ sec. in middelsnelle tot 225 nanoseconde in heel snelle computers. Delen is wat langzamer.

4.4. Logische opdrachten

De logische optelling geeft voor twee binaire woorden als resultaat de "bitsgewijze" som met de afspraak $1+1=1$. Dus $101101' + 001010 = 101111$.

De logische vermenigvuldiging geeft voor twee binaire woorden als resultaat het "bitsgewijze" product. Dus $101101' \times 001010 = 001000$.

Bij de logische vermenigvuldiging kan men de ene operand interpreteren als een masker waardoor men de andere bekijkt. Wanneer de bits van het laatstgenoemde patroon bijv. weergeven dat bepaalde condities al of niet vervuld zijn (1 voor wel en 0 voor niet), bijv. in een register van persoonsgegevens bij de politie kan men de computer aldus als het ware in een oogopslag laten constateren of minstens één van de door het masker zichtbare subset der condities vervuld is: het resultaat van de logische vermenigvuldiging is dan niet 0. En men kan ook heel snel constateren of ze allemaal vervuld zijn door van het resultaat het masker-patroon af te trekken en op 0 te testen. Etc.

Ook is de logische vermenigvuldiging heel geschikt om uit een woord van 32 bits dat 4 karakters van 8 bits bevat een bepaald karakter te lichten.

4.5. Schuifopdrachten

Ook deze opdrachten vindt men in ECOL niet en we zullen ook niet trachten ze in ECOL te omschrijven. Er zijn weer vele typen. Ze verschuiven het hele bitpatroon in een aangegeven register een aangegeven aantal plaatsen naar links of naar rechts en vullen de leeggekomen plaatsen met nullen of enen of met het achterste bit van de oorspronkelijke inhoud; of wat er links uitschuift

komt er rechts weer in (rondschiiven); of het zojuist gereleveerde vindt plaats met de hele bittrein die 2 met name genoemde registers tezamen vormen. Etc.

Deze opdrachten vervullen een duidelijke functie wanneer men een binair getal met een macht van 2 wenst te vermenigvuldigen. Of wanneer bijv. de bits 8-15 een karakter voorstellen dat men op de plaatsen 0-7 wenst te krijgen. Ook bij het in- en uitpakken van meerdere kleine getalletjes in één woord (zie eind van 3).

Een voorbeeld van samengaan van logische en schuifopdrachten levert het vliegtuigreserveringsprobleem. Stel dat de raamplaatsen genummerd zijn 0-19, de eraan grenzende plaatsen 20-39 en dat het bezet zijn van plaats i gekenmerkt wordt door het 1 zijn van de i-de bit in een bepaald woord. Dan kan men met slechts een paar opdrachten constateren of er 2 plaatsen naast elkaar vrij zijn: tel het betreffende woord logisch op bijzijn 20 plaatsen naar rechts verschovene. Iedere 0 in het resultaat geeft een paar vrije plaatsen aan. Zijn er geen vrije paren dan bestaat het resultaat uit uitsluitend enen en dat kan men met één opdracht testen. Na een dergelijke test weet men dus of er vrije paren zijn. Om te weten welk paar vrij is is het handig als men over een zgn. normeeschuif-opdracht beschikt die ze schuift het woord zoveel plaatsen op dat de eerste 0 (als het woord met een 1 begint) of de eerste 1 (als het woord met een 0 begint) op kop komt en deponeert het aa. tal plaatsen waarover geschoven is in een register.

4.6. Sprongopdracht

De ECOL opdracht "NAAR 27" vindt hier zijn analogon in "NAAR $j+R_K$ ": ga verder met de opdracht die in geheugenplaats $j+R_K$ zit. Dit herinnert ons eraan dat we wel hebben gezegd dat de opdrachten ook in het geheugen zitten, maar niet in welke volgorde ze worden afgewerkt. Het ligt voor de hand, dat de normale volgorde is de volgorde waarin ze in het geheugen zitten. Een opdracht als "NAAR $j+R_K$ " doorbreekt dit.

Hiernaast heeft men dan veelal nog opdrachten als NAAR $M(j+R_K)$ (strooisprong), waarmee men aangeeft dat in geheugenplaats $j+R_K$ het adres te vinden is waar moet worden verder gegaan. D.w.z. NAAR $M(10)$ is hetzelfde als NAAR 88 wanneer geheugenplaats 10 het getal 88 bevat.

De ECOL opdracht ALS...DAN...ANDERS..., waarmee men dus twee plaatsen moet aangeven waar kan worden verder gegaan, treft men bij computers niet aan. In plaats daarvan heeft men een opdracht ALS...DAN... Wanneer de conditie vervuld is wordt verder gegaan op de aangegeven plaats. Wanneer de conditie niet vervuld is wordt gewoon verder gegaan op de geheugenplaats volgende op die waar de opdracht ALS...DAN... zelf staat.

Achter ALS... mag natuurlijk geen ingewikkelde toestand staan zoals in ECOL. De opdrachten zijn veelal van het type

ALS $R_1 > 0$ DAN $j+R_K$

ALS $R_1 = 0$ DAN $j+R_K$

e.d.

Deze opdrachten passen weer in het standaardformaat (zie 4.1), waarbij het typenummer in 24-31 tot uitdrukking brengt of de conditie >0 of $=0$ o.i.d. luidt.

4.7. Subroutine sprong

Dit is een sprongopdracht met hetzelfde gevolg als de NAAR opdracht, met nog iets extra's. Hij bergt namelijk op een aan te geven plaats het adres (of het adres + 1) op waar hij zelf staat.

Als de opdracht bijv. luidt

SUB $j+R_K, R_1$

en hij staat zelf op adres 723, dan vindt een sprong plaats naar adres $j+R_K$ en het getal 723 (resp. 724) wordt opgeborgen in R_1 . De opdracht past blijkbaar nog steeds in het standaardformaat.

We illustreren het belang van de subroutinesprong aan een voorbeeld. Stel dat men in een programma op vele plaatsen een sinus wenst uit te rekenen. Men zal het dan op prijs stellen slechts eenmaal een stuk programma te hoeven schrijven dat de sinus bepaalt van het argument dat bijv. in R_1 zit; telkens als men in het eigenlijke programma een sinus wil laten uitrekenen stopt men het argument in R_1 en schrijft een sprongopdracht naar het stuk sinus programma. Het probleem zit nu hierin, dat men na het sinusprogramma doorlopen te hebben weer wenst terug te keren naar de plaats waar men vandaan gekomen is, maar dit is telkens ver-

schillend (zie figuur 1):

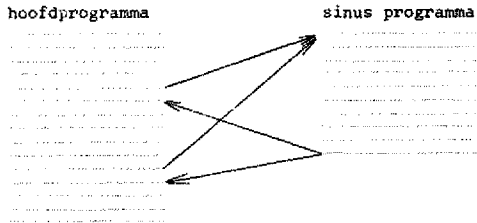


fig. 1

Door naar het sinusprogramma te springen met een subroutine-sprong die het adres waar hij zelf staat (of dit adres + 1) opbergt bijv. in R_6 en de sinusroutine te laten eindigen met NAAR $1+R_6$ (resp. NAAR R_6) is men uit de moeilijkheden. Een zo eindigend stuk programma noemt men een subroutine.

Het is ook fijn wanneer naast de opdrachten SUB $j+R_k$, R_i en SUB $M(j+R_k)$, R_i nog subroutine-sprongen bestaan die het terugkeeradres niet in R_i opbergen, maar in het adres aangewezen door R_i . We zullen deze opdrachten aanduiden met SUBS $j+R_k$, R_i , resp. SUBS $M(j+R_k)$, R_i .

8. Stapelprogrammering

Een faciliteit waarvan vooral de geavanceerde programmeur plezier beleeft is dat een bepaald register, bijv. R_7 de eigenschap heeft dat telkens wanneer in een opdracht de uitdrukking $M(j+R_7)$ voorkomt de volgende extra actie plaatsvindt: als er iets uit de geheugenplaats $j+R_7$ wordt afgelezen wordt R_7 voor dat aflezen met 1 verlaagd, als er iets wordt opgeborgen wordt R_7 na dat opbergen met 1 verhoogd.

Voorbeelden

(a) $R_7 := 81$; $R_8 := R_7 + M(R_7)$. Effect: $R_7 := 81$; $R_8 := 80$; $R_8 := R_8 + M(80)$
 (b) $R_7 := 81$; $M(R_7) := R_8$. Effect: $R_7 := 81$; $M(81) := R_8$; $R_7 := 82$
 (c) $R_7 := 81$; $M(R_7) := M(R_7) + R_8$. Effect: voor de bepaling van het rechterlid wordt eerst R_7 met 1 verlaagd, zodat het rechterlid dus het effect $M(80) + R_8$ heeft; dit wordt opgeborgen in de geheugenplaats met adres 80 waarna R_7 weer met 1 verhoogd wordt. Dus totaal effect: $M(80) := M(80) + R_8$; $R_7 := 81$.

Deze faciliteit maakt het mogelijk het geheugen als een zgn. stapel te beschouwen, waarbij R_7 als de stapelwijzer fungeert. Dit is van belang als men een hoeveelheid informatie wenst op te slaan die men in omgekeerde volgorde wenst te gebruiken. (Voorbeelden hiervan worden elders in deze cursus gegeven.) Men legt dan de op te slaan informatie als het ware op een stapel, elke nieuwe informatie-eenheid er boven op en bij het terughalen van de informatie neemt men telkens de bovenste informatie-eenheid eraf (denk aan een stapel kaarten). In dit licht bezien wijst R_7 dus steeds naar de eerste vrije plaats van de stapel.

Heel prettig is het terugkeeradressen van subroutines op een stapel te kunnen leggen. Dit kan blijkbaar door de subroutine aan te roepen met SUBS ..., R_7 , waarbij analoog geldt dat R_7 dan na dat opbergen met 1 verhoogd wordt.

5. Werking van een computer in principe

Het gebeuren in de computer speelt zich af in twee fasen, die elkaar voortdurend afwisselen.

Fase 1: de aan de beurt zijnde opdracht wordt uit het geheugen afgelezen.

Fase 2: de afgelezen opdracht wordt uitgevoerd.

Deze afwisseling van fase 1 en 2 noemt men de basiscyclus.

Het besturingsorgaan van de computer bevat daartoe twee registers: de opdrachtwijzer (OW) houdt het adres bij waar de volgende opdracht vandaan gehaald moet worden; het opdrachtregister (OR) bevat de opdracht die uitgevoerd moet worden.

Het hele gebeuren in de computer berust er nu op, dat op het juiste tijdstip de juiste elektrische verbindingswegen bestaan.

In fase 1 moet OW verbonden zijn met de zgn. selectie-ingang van het geheugen, d.i. die ingang die bepaalt, welk adres wordt uitgelezen en de uitgang van het geheugen moet verbonden zijn met OR (zie figuur 2).

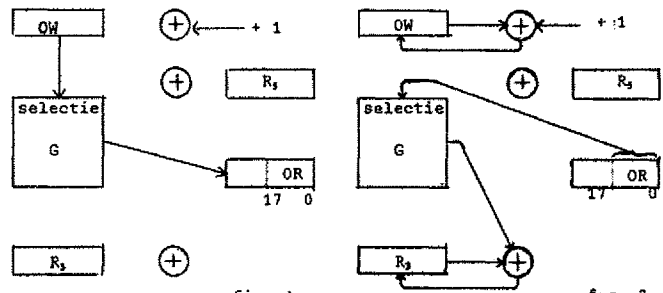


fig. 2

We zullen de selectie-ingang steeds bovenaan tekenen, de ingang waar men de op te bergen informatie aanbiedt links en de uitgang van het geheugen rechts.

In fase 2 wordt OW met 1 verhoogd. Stel dat we de eenvoudige opdracht $R_8 := R_8 + M(R_7)$ hebben. Dan moeten de bits 0-17 van OR met de selectie-ingang van het geheugen verbonden worden. De uitgang van het geheugen wordt, samen met de uitgang van R_8 , aangeboden aan een opteller. De uitgang van de opteller moet verbonden zijn met de ingang van R_8 (zie figuur 3).

Voor een opdracht als $R_8 := R_8 + M(80 + R_8)$ wordt fase 2 afgebeeld in figuur 4.

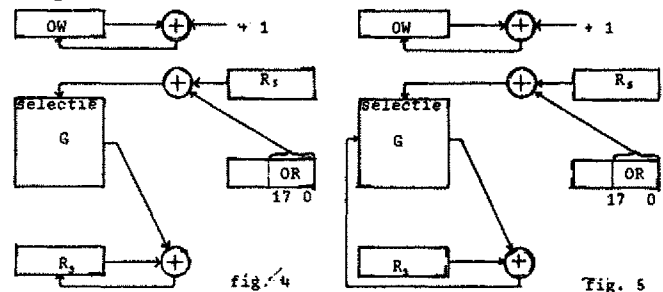


fig. 4

Figuur 5 geeft fase 2 weer voor $M(80 + R_8) := M(80 + R_8) + R_8$.

Sprongopdrachten van het type NAAR 80 worden eenvoudig gerealiseerd door 0-17 van OR over te nemen in OW (zie figuur 6).

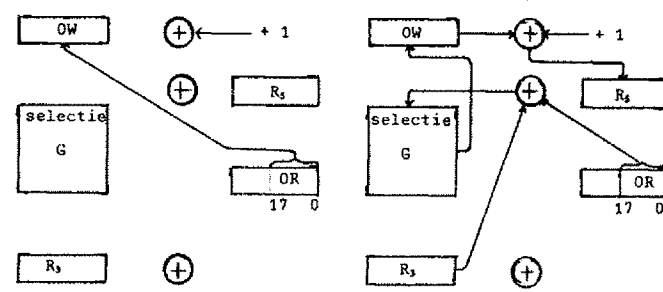


fig. 6

fig. 7

In figuur 7 ten slotte ziet men de opdracht SUB $M(80 + R_8)$, R_8 .

Het is duidelijk dat er in dit betoog wat vereenvoudigingen zijn aangebracht. Elk der fasen zal uit meerdere delen moeten bestaan. Bijv. in figuur 3 zal men moeten voorkomen dat nadat de met 1 vermeerderde OW is teruggeschreven in OW deze nieuwe waarde nog eens met 1 vermeerderd zou worden en nog eens. Analoog voor het bij elkaar tellen van geheugeninhoud en R_8 .

6. Praktische realisatie

6.1. Geheugen

Door elk ferrietringetje lopen 3 draden (zie figuur 8), een "horizontale" draad, een "verticale" draad en een leesdraad.

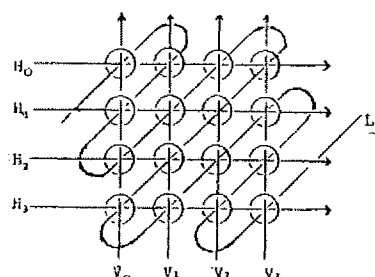


fig. 8

De vierkante hystereselus van de ringetjes (figuur 9) maakt dat er een stroomsterkte I_0 bestaat zodanig dat een stroom van deze sterkte niet bij machte is een ringetje van magnetische richting te doen omslaan, terwijl een stroom van dubbele sterkte dit wel kan. Stuur men bijv. door H_1 en V_1 stromen met sterkte I_0 in de richting van de pijlen of juist in tegengestelde richting, dan neemt het ringetje op hun kruispunt een magnetisatie richting aan passend bij de stroomrichting en alle andere ringetjes blijven staan.

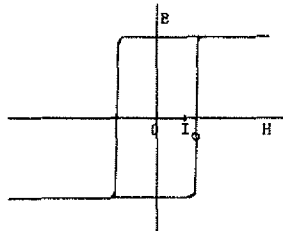


Fig. 9

Wenst men later te weten hoe dit ringetje staat dan stuurt men een stroom door H_1 en V_1 in de richting der pijlen. Past de magnetisatorichting van het ringetje bij de stroomrichting, dan blijft het staan en wordt er een andere spanning in de leesdraad geïnduceerd dan wanneer het niet paste en dus omklapte.

Het uitlezen is blijkbaar destructief: na afloop past het ringetje beslist bij stromen in de richting van de pijlen. Er moet derhalve een voorziening getroffen worden om zo'n uitgelezen ringetje meteen weer goed te zetten.

Op de vraag hoe nu juist H_1 en V_1 stroom gaan voeren als wij dat wensen gaan we straks in.

6.2. Flip-flop

Ook een flip-flop is een geheugen element, maar dan een elektronisch. Het is een kastje met 2 ingangen en 2 uitgangen (fig. 10). Van deze uitgangen heeft er steeds één een "hoge" spanning (die we verder als 1 zullen aanduiden) en één een "lage" spanning (verder aan te duiden als 0). Als beide ingangen 0 zijn verandert er niets. Als een der beide ingangen 1 gemaakt wordt, wordt de bijbehorende uitgang 1 (en dus de andere 0). Het is verboden beide ingangen 1 te maken. Een flip-flop is dus net een wip: die "onthoudt" ook aan welke kant hij het laatst naar beneden gedrukt is, ook als die druk wordt opgeheven. Maar als men aan beide kanten tegelijk drukt gebeurt er iets onvoorspelbaars.

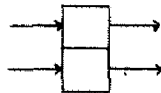


fig. 10

We zullen bovendien afspreken dat we zeggen dat de flip-flop een 1 voorstelt als de onderste uitgang de waarde van 1 heeft.

6.3. Poorten

We hebben verder nog elektronische schakelaars nodig om op commando verbindingswegen te openen of af te sluiten. Deze functie wordt door poorten verricht.



fig. 11



fig. 12

Poorten hebben een aantal ingangen en één uitgang. Bij de plus poort (or gate) (fig. 11) is de uitgang slechts dan 1 als alle ingangen 1 zijn. Bij de maal poort (and gate) (fig. 12) is de uitgang slechts dan 1 als alle ingangen 1 zijn.

De principe schema's van poorten zijn simpel (fig. 13 en 14).

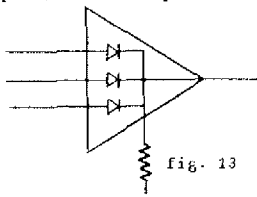


fig. 13

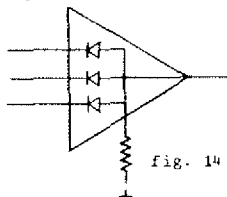


fig. 14

De werking wordt verklaard door op te merken dat een diode links nooit een hogere spanning kan hebben dan rechts.

6.4. Verbindingswegen

Een verbindingsweg tussen bijv. de uitgang van G en de ingang van OR die alleen actief is in fase 1 (zie fig. 2) komt nu als

volgt tot stand. Laat fase 1 daardoor gekenmerkt zijn dat op één heel bepaalde draad, verder te noemen F1, het signaal 1 staat idem fase 2 een draad F2.

Het geheugen heeft voor elke bit van het op te leveren woord een uitgang en deze uitgang is doorgaans dubbel uitgevoerd in die zin dat voor elk bit ook de tegengestelde waarde (d.w.z. 0 wordt 1 en andersom) is af te lezen.

Dan bouwt men tussen de geheugenuitgang en de ingangen van de flip-flops waaruit OR bestaat het volgende circuit, even vaak herhaald als er bits in een woord zijn (fig. 15).

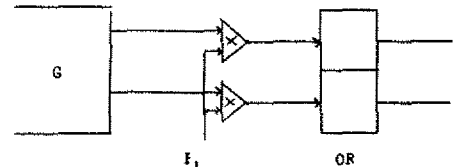


fig. 15

In fase 2 is de zaak duidelijk ingewikkelder, want dan wil men dat een bepaalde verbindingsweg precies dan actief is als er bij een heel bepaald bitpatroon in 24-31 van OR zit.

Het aanwezig zijn van dat heel bepaalde bitpatroon kan men als volgt constateren met een poortschakeling. Ter vermindering van het tekenwerk doen we evenwel maar net alsof we slechts met bits 24-27 van OR te maken hebben. Dan geeft de maal poort in figuur 16 precies dan het signaal 1 af als 27-24 van OR het patroon 0101 voorstellen en bovendien F2 de waarde 1 heeft.

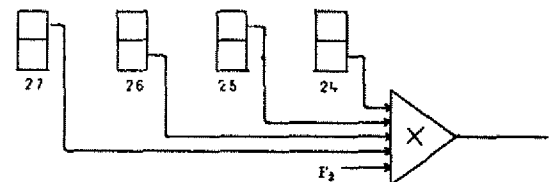


fig. 16

Met het uitgangssignaal van deze poortschakeling kan men nu verbindingswegen activeren analoog aan fig. 15.

Op soortgelijke wijze activeert ook het geheugen de juiste draden H_1 en V_1 (zie fig. 8).

6.5. In- en uitvoer

Het is nu ook wel duidelijk hoe de informatie uit een ponskaart in de computer zou kunnen doordringen: men voert eenvoudig een opdracht in die een verbindingsweg tot stand brengt tussen de uitgangen van de kaartlezer (waar in de vorm van spanningen op draden het gaatjesbeeld vast ligt) en bijv. een bepaalde geheugen plaats. Analoog voor uitvoer.

7. Programmaverwerking

Wat gebeurt er nu als men een programma ter verwerking aanbiedt (bijv. een ALGOL 60 programma). In de computer is "permanent" een groot programma aanwezig, ALGOL compiler geheten. Met een druk op de knop forceert de operator (= bediener van de machine) een sprongopdracht naar dit programma, zodat het aan de gang gaat. Dit programma leest nu de programmatext karakter voor karakter in (zie 6.5) en slaat het als tekst op in het geheugen. Deze tekst is nu vanzelfsprekend voor de computer nog geen programma (vgl. 4 hoe computeropdrachten er in het geheugen uitzien). De ALGOL compiler gaat nu (op een hoogst ingewikkelde manier met deze tekst manipuleren en "berekenen" een voor een de bitpatronen van de opdrachten die de computer straks zal moeten gaan uitvoeren om het door de auteur van het programma gewenste effect te bewerkstelligen. Zo zal de compiler moeten uitmaken welke geheugenplaats hij aan de variabele piet zal toekennen en hij zal expressies als $a := b \times c + d \times \sin(x)$ moeten reduceren tot de simpele opdrachten van 4 (incl. een subroutine aanroep voor de functie $\sin(x)$, welke subroutine vast in de ALGOL compiler is opgenomen).

De bedoelde bitpatronen schrijft hij netjes in volgorde in het geheugen en als hij daarmee helemaal klaar is eindigt hij met een sprongopdracht naar het eerste van deze bitpatronen, waarmee dan het aldus vertaalde programma geactiveerd (we zeggen ook geëxecuteerd) wordt.

Hiermee hebben we een passant een zeer belangrijke eigenschap van een computer leren kennen: omdat de bitpatronen van opdrachten niet "anders" zijn dan die van getallen en teksten kunnen bitpatronen die op een zeker moment uitkomst zijn van een berekening later als opdrachten optreden.

Dit is ook van belang bij het beantwoorden van de vraag hoe de ALGOL compiler zelf in de machine is gekomen en algemener hoe het "eerste" programma in de machine komt. Gewoonlijk wordt van fabriekswege op onuitwisbare wijze een uiterst simpel programmaatje in het geheugen geplaatst van maar een paar opdrachten dat bijv. alleen in staat is zgn. binaire programma's te lezen, dat zijn programma's die helemaal in enen en nullen uitgeschreven zijn precies zoals ze in het geheugen moeten komen. Om een programma zo uit te schrijven is natuurlijk niet leuk, vandaar dat men dat maar met één programma doet, nl. een programma dat in staat is programma's in een prettigere code in te lezen. In die prettigere code schrijft men dan wellicht een inleesprogramma voor een nog hoger geordende code.

8. In- en uitvoer; verkeer met achtergrondgeheugen

Het is vrijwel nooit doelmatig een gebruikersprogramma zelf direct met in- en output organen of achtergrondgeheugen te laten communiceren. Daarvoor is een hedendaagse computer eenvoudig te snel. Als men in ECOL schrijft a:= LEES, en pas op dat moment zou een ponsbandlezer of ponskaartlezer karakter voor karakter van band of kaart gaan lezen dan zou deze opdracht vele milliseconden in beslag nemen en dat terwijl de computer slechts microseconden nodig heeft om een rekenoperatie uit te voeren. Bij verkeer met magneetband, -trommel of -schijf is het al niet anders. Men staat daar eerst een tijd te wachten (misschien minuten bij magneetband, tientallen milliseconden bij trommel of schijf), voordat de juiste geheugenplaats langs lees- of schrijfkop komt, maar ook als deze plaats gearriveerd is en ook als men dan een aantal opeenvolgende geheugenplaatsen op dat medium wenst te bespelen, dan zijn deze apparaten nog steeds "langzaam": ergens tussen 10000 en 100000 woorden per seconde.

Moderne, snelle computers beschikken daarom over een of meer zgn. communicatiesatellieten. Dat zijn eigenlijk kleine computertjes met een eigen besturingsorgaan en soms ook nog met een eigen klein geheugentje; soms herbergen deze één vast (door de fabriek erin geplaatst) programma, soms zijn ze nog programmeerbaar ook. De opdrachtenset is evenwel geheel gericht op communicatie met de randapparatuur en de eigenlijke computer (verder te noemen de centrale computer).

Het programma in de centrale computer kan nu aan de communicatiesatelliet opdracht geven om bijv. 1000 woorden uit het trommelgeheugen af te lezen, op een bepaalde plaats te beginnen en deze in het werkgeheugen van de centrale computer te plaatsen, op een bepaalde plaats te beginnen en na afloop te melden dat het gebeurd is.

Dit werpt het boeiende probleem op twee computers, die elk in hun eigen ritme werken, met elkaar te laten samenwerken. Omdat deze samenwerking o.a. de bedoeling heeft dat de ene computer in het geheugen van de andere leest of schrijft zal er in elk geval een soort voorrangsregeling moeten zijn die uitmaakt wie toegang krijgt tot het geheugen als ze allebei tegelijk komen.

Fundamenteel is evenwel dat de centrale computer aan de satelliet een seintje moet kunnen geven dat deze een karweitje moet opknappen en dat de satelliet computer aan de centrale computer moet kunnen berichten dat het karweitje klaar is.

Dat ze in elkaars geheugen kunnen lezen en schrijven is daarvoor niet voldoende. Het zou immers te tijdrovend zijn (en te vervelend) het computerprogramma voortdurend naar bepaalde geheugenplaatsen te laten kijken om te zien of daar al wat staat. Dat zou net zo iets zijn als een telefoon zonder bel: men zou elke keer naar de telefoon moeten lopen om te horen of er soms iemand iets wil zeggen. De bel dient ervoor om aan te kondigen dat iemand contact wenst.

De computers moeten dus elkaars aandacht kunnen trekken. We schetsen hoe de satelliet een seintje geeft aan de centrale computer (andersom gaat net zo of nog eenvoudiger, omdat de satelliet mogelijk wel tijd heeft om voortdurend naar bepaalde geheugenplaatsen te kijken).

Als de satelliet een seintje wil geven aan de centrale computer zet hij op een bepaalde draad, de interrupt-draad, het signaal 1.

De centrale computer let in fase 1 van de basiscyclus op de interruptdraad. Als daar het signaal 0 op staat gaat alles zoals in 6 beschreven. Als er evenwel het signaal 1 op staat wordt de selectie ingang van het geheugen niet verbonden met OT, maar wordt er een bepaald bitpatroon opgezet. Stel dat dit het bitpatroon 00....011000 is. Dan wordt dus geheugenplaats 24 uitgelezen en de inhoud daarvan wordt geplaatst in OR. Wat er nu verder gebeurt hangt er natuurlijk van af wat dit voor een opdracht is, maar het is wel duidelijk dat het op dat moment werkende programma ruwelijk onderbroken is. Dit is op zichzelf genomen hoogst bedenkelijk, want de interrupt van de satelliet komen op volstrekt onvoorspelbare momenten en het is doorgaans heel ongewenst dat een werkend programma zo maar ergens onderbroken wordt.

Gewoonlijk evenwel zal de programmeur van het zgn. bedrijfsstelsel (dat is een groot vast programma dat altijd in de computer zit, net als de ALGOL compiler) op de betreffende geheugenplaats 24 een subroutinesprong geplaatst hebben naar een stuk programma waar verder kennis genomen wordt van de mededelingen die de satelliet computer in het geheugen van de centrale computer heeft geplaatst. Omdat we met een subroutinesprong naar dit stuk programma zijn gesprongen, is het adres waar het onderbroken programma gebleven was, bewaard gebleven en kan daarheen later worden teruggekeerd.

Gewoonlijk zijn er nog voorzieningen om interrupts, al naar hun herkomst, met verschillende prioriteit af te werken en ook om al naar deze herkomst een andere geheugenplaats dan 24 te laten aflezen, alles ter verhoging van de snelheid.

Het interrupt-mechanisme maakt het mogelijk om een nieuw programma met zijn gegevens alvast in het computergeheugen te laten opnemen, terwijl een ander programma nog bezig is. En ook om de output van een programma, dat veel meer regeldruk nodig heeft dan rekentijd (een zgn. output intensief programma) op de regeldrukker te produceren terwijl een nieuw programma al weer bezit van de centrale computer genomen heeft.

Het interrupt-mechanisme is ook de grondslag van de mogelijkheid tot multiprogramming, d.w.z. dat de computer schijnbaar aan meerdere programma's tegelijk rekent, maar in werkelijkheid het ene programma laat werken in de tijden dat het andere niet verder kan omdat het bijv. op informatie van een achtergrondgeheugen staat te wachten.

Bij grote computersystemen komt ook steeds meer multiprocessing in zwang, d.w.z. dat het computersysteem meerdere centrale computers bevat die elkaar ook weer via een interruptsysteem signaleren.

Tenslotte maakt het interruptsysteem timesharing mogelijk, d.w.z. dat vele gebruikers om de beurt een kort stukje rekentijd toegemeten krijgen. Deze mensen zijn dan via een terminal (doorgaans een schrijfmachine-achtig apparaat, soms op grote afstand van de computer geplaatst en daarmee via telefoonlijnen verbonden) met de computer verbonden en geven over deze terminal hun opdrachten aan de computer en krijgen daarop ook hun antwoorden.

Een heel belangrijk voordeel van timesharing is de mogelijkheid die het geeft programma's in samenspraak met een computer te testen en te verwerken. Bij de normale wijze van programmaverwerking (in batch, zoals dat heet) kost het dagen om een programma een aantal keren, in een steeds wat andere versie, aan te bieden. Met timesharing kost hetzelfde slechts uren of kwartieren.

Dat al het geïnterrupteerd, dat op deze bladzijden beschreven werd, goed verloopt is dankzij het bedrijfsstelsel, dat boekhoudt waar de informatie betreffende elk programma zich bevindt, zoekt welk programma aan de beurt is en nog veel meer. Hierover wordt elders in deze cursus gerapporteerd.

9. Allerlei randapparatuur

9.1. Ponsbandlezer

200 - 2000 tekens per seconde. Werkt foto-electrisch of capacitief. 5, 7 of 8 sporen.

9.2. Ponsbandponser

100 - 200 tekens per seconde. 5, 7 of 8 sporen.

9.3. Ponskaartlezer

10 - 20 kaarten per seconde. Werkt met contactborstels of foto-electrisch.

9.4. Ponskaartponser

100 - 500 kaarten per minuut.

9.5. Magneetbandapparaat

Rollen magneetband met maximale lengte 720 meter. Snelheid ca. 3 m/sec., start/stoptijd enkele milliseconden. 10 - 60 tekens van 6 of 8 bits (+ parity) per mm. Dus per rol enkele tientallen miljoenen tekens. Overdrachtsnelheid in de orde van 100.000 tekens per seconde.

9.6. Magneettrommel

Ronddraaiende magnetiseerbare cilinder met enkele honderdduizenden tot enkele miljoenen woorden. Rotatiesnelheid 20 - 100 maal per seconde, dus wachttijd in de orde van tientallen milliseconden. Overdrachtsnelheid plm. 100.000 woorden per seconde. Terugdenkend aan de vergelijking in het begin van 1 (kaartenbak op tafel versus archief op zolder) is de zolder wel ver weg: om iets uit het archief te halen duurt 10000 maal zo lang als om iets uit de kaartenbak te halen, d.i. een etmaal versus 8 seconden.

9.7. Magneetschijven

Stapel ronddraaiende magnetiseerbare schijven. Werkwijze vergelijkbaar met die van trommel. Elke schijfkant heeft zijn eigen lees- schrijfkop die in radiale richting instelbaar is. Dit kost wat extra tijd (± 100 milliseconden). Voordelen boven magneettrommel: schijven zijn uitwisselbaar; bij gelijke geheugenomvang minder duur.

9.8. Magneetkaarten

De informatie staat op magnetiseerbare kaartjes (maten ca. 35 x 9 cm). Per kaart ca. 400.000 tekens. In de tijd van $\pm 0,2$ sec. wordt de gewenste kaart uit het magazijn gelicht, langs een draaiende trommel gevoerd en aldus afgelezen.

9.9. Regeldrukker

10 - 20 regels van maximaal ± 150 tekens per seconde. Twee uitvoeringen (in beide wordt het papier horizontaal bedrukt).

- (a) horizontale cilinder met op elke drukpositie een hele tekenset (dus max. 150 keer) draait 10 of 20 maal per seconde in het rond en als het juiste teken voordraait slaat een hamer het papier met carbonlint en al tegen de cilinder.
- (b) ketting met 5 tekensets draait horizontaal voor het papier langs en telkens als het juiste teken op de juiste plaats staat slaat een hamer het papier met carbonlint en al tegen de ketting (allebei fraaie staaltjes van techniek: het tijdstip van afslaan van de hamer mag niet meer dan 50 μ sec. variëren op straffe van dansende of wiebelende letters).

9.10. Plotter

Dit is een tekenapparaat. Het papier zit met pennen vast op een rol die heen en weer kan draaien in stapjes van 0,1 mm. In axiale richting loopt hierboven een wagentje met een schrijfpenn, welk wagentje kan bewegen in stapjes van 0,1 mm. Het apparaat kent als opdrachten:

- (a) haal pen van papier
- (b) zet pen op papier
- (c) draai rol 1 stapje vooruit
- (d) draai rol 1 stapje achteruit
- (e) pen 1 stapje naar links
- (f) pen 1 stapje naar rechts.

De opdrachten (c of d) en (e of f) kunnen ook tegelijk worden gegeven, zodat de pen in feite in de 8 hoofdwindrichtingen over het papier kan bewegen. Het apparaat doet 300 stapjes per seconde (moderne uitvoering is nog sneller).

9.11. Visual display

Een soort televisiescherm waarop de computer een beeld kan projecteren. Heel interessant is dat men dit apparaat ook als input apparaat kan gebruiken. Bij het apparaat hoort een potloodachtig apparaatje aan een snoer, lichtpen geheten, voorzien van een lichtgevoelige cel.

De computer projecteert op een bepaalde plaats op het scherm een punt. Houdt men de lichtpen bij deze punt en beweegt men hem er vervolgens kalmpjes vandaan dan constateert de computer de vermindering van lichtsterkte en verschuift de punt net zo lang totdat de lichtsterkte weer maximaal is. Zo kan de computer dus altijd weten waar de lichtpen zich bevindt, met andere woorden de beweging van de pen kan gebruikt worden om gegevens in de computer te voeren.

5.2 DATASTRUCTUREN

Hieronder volgt de tekst van de lezing over datastructuren, zoals die gehouden is op een heroriënteringscursus computerkunde door Drs. C.A.Ch. Górts.

1. Inleiding

De informatie waarop een programma moet werken bezit vaak een zekere structuur. Voordat men het programma gaat schrijven moet men zich van deze structuur bewust zijn en zich afvragen op welke manieren men deze structuur in het geheugen van de computer kan representeren. Bij het maken van een keuze uit deze manieren moet men zich goed voor ogen houden wat men met de informatie wil doen. In enkele voorbeelden zal dit worden toegelicht.

2. De woordenlijst

Een in de literaire wetenschap veel voorkomende bezigheid, waarvoor tegenwoordig vaak computers worden gebruikt, is het opstellen van een woordenlijst van een tekst (een boek, of het hele werk van een auteur): deze woordenlijst bestaat in zijn eenvoudigste vorm uit een alfabetische lijst van alle woorden die in de tekst voorkomen.

De computer leest de tekst woord voor woord. Van ieder woord wordt nagegaan of het al in de woordenlijst voorkomt. Zo niet, dan wordt het eraan toegevoegd.

Het gaat er nu dus om de woordenlijst in een handige vorm in het geheugen op te bouwen, uitgaande van de volgende drie werkzaamheden die we op de woordenlijst willen uitvoeren: het opzoeken of een woord voorkomt, het toevoegen van een woord en het (na afloop) in alfabetische volgorde afdrukken van de woordenlijst.

Eenvoudigheidshalve zullen we aannemen dat de woorden uit niet meer dan 10 letters bestaan. Voor een woord reserveren we 10 plaatsen in een rij a. Als het woord minder dan 10 letters heeft vullen we de resterende plaatsen op met nullen. We nemen aan, dat bekend is hoe men van twee woorden kan uitmaken welk het eerst komt in alfabetische volgorde.

2.1. Sequentieel

Uit § 20 van computerkunde is duidelijk dat het niet verstandig is de woorden van de woordenlijst in volgorde van binnenkomst in een rij te zetten: om op te zoeken of een woord voorkomt moet men dan de hele woordenlijst langslopen.

Het opzoeken of een woord voorkomt gaat veel sneller als de woordenlijst alfabetisch geordend is: dan kan de halveringsmethode worden toegepast. (Computerkunde, § 20, 4.)

RIJ a	
1	e n 0 0 0 0 0 0 0 0 0
11	h e t 0 0 0 0 0 0 0 0 0
21	h o e 0 0 0 0 0 0 0 0 0
31	v e r 0 0 0 0 0 0 0 0 0
41	w a r m 0 0 0 0 0 0 0 0 0
51	w a s 0 0 0 0 0 0 0 0 0
61	

figuur 1

Als we echter de woorden sequentieel, d.i. in alfabetische volgorde achter elkaar (zie figuur 1, het begin van de tekst is: "hoe warm het was en hoe ver") opbergen, hebben we het bezwaar dat het toevoegen van een woord aan de woordenlijst te veel tijd kost: er moet een plaats worden vrijgemaakt door een deel van de rij op te schuiven.

2.2. De keten

We willen dus de woordenlijst met behoud van de alfabetische volgorde in het geheugen representeren, maar zo, dat het mogelijk is snel een woord toe te voegen. Dit kan als we het doen in de vorm van een keten (zie figuur 2).

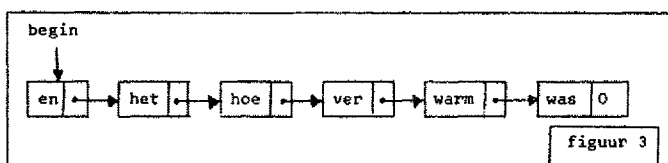
RIJ a	
1	h o e 0 0 0 0 0 0 0 0 56
12	w a r m 0 0 0 0 0 0 0 0 34
23	h e t 0 0 0 0 0 0 0 0 1
34	w a s 0 0 0 0 0 0 0 0 0
45	e n 0 0 0 0 0 0 0 0 23
56	v e r 0 0 0 0 0 0 0 0 12
67	

begin=45
vrij=67

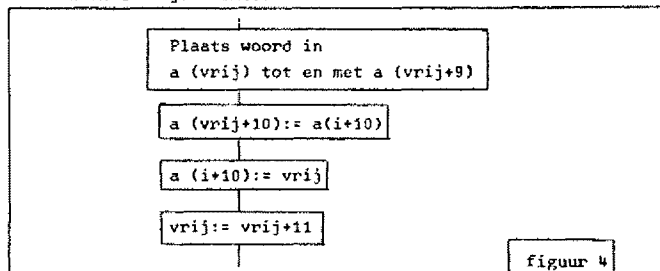
figuur 2.

De woorden staan nu niet in alfabetische volgorde in de rij, maar aan ieder woord is een getal toegevoegd (de link) dat aangeeft waar men het volgende woord kan vinden. Bovendien zijn er twee variabelen, die we begin en vrij noemen, die aangeven waar het eerste woord staat en welke de eerste vrije plaats in de rij is. Het alfabetisch laatste getal heeft de link 0.

Vaak gebruikt men voor een keten (omdat het niet relevant is op welke index een woord te vinden is) de notatie van figuur 3, met links aangeduid door pijlen.



Het toevoegen van een woord aan de lijst is nu eenvoudig een kwestie van het bijwerken van enkele links. Het woord zelf plaatsen we achteraan op de vrije plaatsen (zie figuur 4, het woord wordt ingevoegd na het woord op index i; er moet natuurlijk een extra voorziening bij voor het geval het nieuwe woord alfabetisch vooraan in de lijst komt).

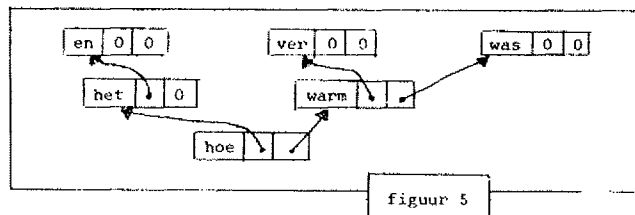


Het toevoegen van een nieuw woord is bij de keten dus eenvoudig. Het opzoeken van een woord is nu echter weer moeilijker: we krijgen immers het middelste woord niet snel te pakken. We zoeken daarom een betere oplossing van ons probleem.

We hebben de keten besproken omdat hij in veel andere toepassingen van nut kan zijn.

2.3. De boom

Om zowel snel te kunnen opzoeken als snel te kunnen toevoegen gebruiken we een boom: vanuit een woord wordt verwezen naar een linker en een rechter tak (deze kunnen leeg zijn). In de linker tak van een woord staan alle woorden die alfabetisch eerder komen, in de rechter tak de woorden die alfabetisch later komen (zie figuur 5).



Algemeen gedefinieerd is een (binaire) boom een verzameling elementen, die leeg is, of bestaat uit een element en twee takken, een linker en een rechter tak, die beide ook weer bomen zijn.

RIJ a	
1	h o e 0 0 0 0 0 0 0 25 13
13	w a r m 0 0 0 0 0 0 0 61 37
25	h e t 0 0 0 0 0 0 0 49 0
37	w a s 0 0 0 0 0 0 0 0 0
49	e n 0 0 0 0 0 0 0 0 0
61	v e r 0 0 0 0 0 0 0 0 0
73	

vrij=73

figuur 6

Ieder woord is nu dus voorzien van twee links (zie figuur 6). De tijd, nodig voor het opzoeken of een woord voorkomt, zal afhangen van de vorm van de boom en dus van de volgorde waarin de woorden binnenkomen. Over het algemeen gaat het sneller dan bij de keten.

Ook het toevoegen van een nieuw woord gaat snel: zet het maar op de vrije plaatsen en werk enkele links bij.

Een speciale moeilijkheid bij het gebruik van een boom is hoe we er voor kunnen zorgen dat de woorden na afloop in alfabetische volgorde worden afgedrukt. (Bij de andere methoden, sequentieel en keten, vormde dit geen probleem.) We moeten de woorden in goede volgorde van de boom plukken: een woord moet worden geplukt na alle woorden in zijn linker tak en voor alle woorden in zijn rechter tak. Vanwege de recursieve definitie van een boom ligt een recursieve procedure voor de hand (zie figuur 7). Door de aanroep plukboom (1) wordt dan de woordenlijst in alfabetische volgorde afgedrukt.

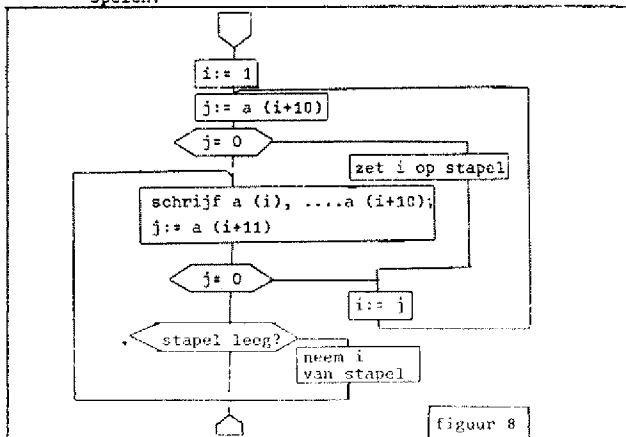
```

procedure plukboom (i); integer i;
begin integer j;
  j:= a[i+10];
  if j # 0 then plukboom(j);
  for j:= 0 step 1 until 9 do PRINT(a[i+j]);
  j:= a[i+11];
  if j # 0 then plukboom(j)
end;

```

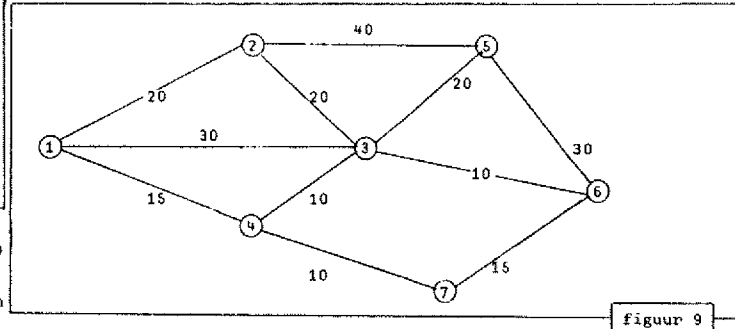
figuur 7

Als we het niet recursief doen moeten we zelf een stapel organiseren, waarin we onthouden langs welke weg we in de boom omhoog klimmen (figuur 8). Voor een goed begrip van dit in volgorde plukken van de woorden van de boom is het wel gewenst deze programma's op een eenvoudig voorbeeld na te spelen.



3. Kortstewegprobleem

Gegeven is een netwerk met punten (genummerd 1,2,...,n) en lijnstukken tussen punten, de lijnstukken hebben gegeven lengten (zie figuur 9). Als punten i en j door een lijnstuk verbonden zijn, zullen we de lengte van dit lijnstuk aangeven met $d(i, j)$.



Gevraagd de kortste weg langs deze lijnen van een gegeven punt t naar een gegeven punt e.

Voordat we ons afvragen hoe we het netwerk in het geheugen zullen representeren gaan we eerst na volgens welke methode we het probleem kunnen oplossen. We doen het zo:

We bouwen een verzameling S op van punten en onthouden van ieder punt $i \in S$ de afstand $a(i)$ tot b. We beginnen met $S = \{b\}$ en $a(b) = 0$.

Om een punt aan S toe te voegen bepalen we $m = \min\{a(i) + d(i, j)\}$, het minimum genomen over alle paren (i, j) met $i \in S$, $j \notin S$, i en j door een lijnstuk verbonden. Als (i_0, j_0) het paar is waarvoor dit minimum wordt aangenomen, dan wordt j_0 toegevoegd aan S, met $a(j_0) = m$.

We gaan hiermee door totdat e aan S wordt toegevoegd. Dan is $a(e)$ de afstand van b tot e. (De kortste weg vindt men aan door van e uit terug te rekenen.)

Nu we de methode kennen weten we wat voor eisen we aan de wijze van representatie in het geheugen moeten stellen: We moeten bij een gegeven punt snel alle punten kunnen vinden die ermee verbonden zijn.

3.1. De matrix

Men kan het netwerk in het geheugen vastleggen in een tweedimensionaal array, $g[1:n, 1:n]$ met

$$g[i, j] = \begin{cases} d(i, j) & \text{als } i \text{ en } j \text{ door lijnstuk verbonden} \\ -1 & \text{als } i \text{ en } j \text{ niet door lijnstuk verbonden.} \end{cases}$$

Dit is voor ons doel minder geschikt. De matrix neemt veel geheugenruimte in beslag en men kan niet snel alle met een punt verbonden punten vinden.

3.2. Sequentieel

Aannemend dat in een punt niet meer dan 5 lijnen samenkomen, reserveren we in een rij g voor ieder punt 10 plaatsen (voor de hoogstens 5 lijnstukken en hun lengten). Overtollige plaatsen worden gevuld met nullen (figuur 10).

RIJ g	
1	2 20 3 30 4 15 0 0 0 0
11	1 20 3 20 5 40 0 0 0 0
21	1 30 4 10 6 10 5 20 2 20
31	1 15 3 10 7 10 0 0 0 0
enz.	

figuur 10

De punten bereikbaar vanuit punt i vindt men als $g((i-1) \times 10 + 1), g((i-1) \times 10 + 3), \dots$

3.3. Sequentieel met access-tabel

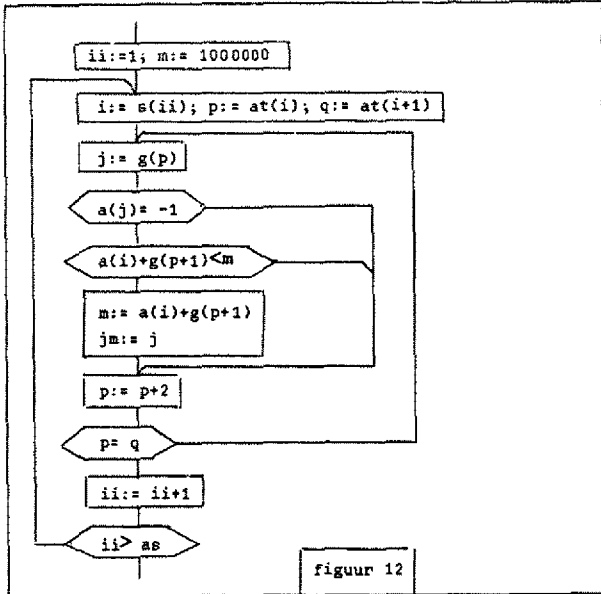
Het ruimteverlies van figuur 9, veroorzaakt doordat niet alle rijen even lang zijn, kunnen we opheffen door alles direct tegen elkaar te schuiven.

RIJ g										
1	2	20	3	30	4	15	1	20	3	20
11	5	40	1	30	4	10	6	10	5	20
21	2	20	1	15	3	10	7	10	2	40
31	3	20	6	30	5	30	3	10	7	15
41	4	10	6	15						

RIJ at										
1	7	13	23	29	35	41	45			

figuur 11

Welk deel van g bij een punt i behoort vindt men met behulp van de accestabel at (figuur 11). Als $at(i) = p$ en $at(i+1) = q$ staan de punten bereikbaar vanuit punt i in g(p) tot en met g(q)-1.



figuur 12

In figuur 12 vindt men, voor het geval dat de gegevens van het netwerk zich in rijen g en at bevinden, de kern van de kortste weg algoritme, namelijk het bepalen van

$$m = \min \{a(i) + d(i, j) \mid i \in S, j \notin S\}$$

Er wordt aangenomen dat een rij s de punten van S bevat (aantal as) en een rij a, de afstanden en wel:

als $i \in S$ is $a(i)$ de afstand van b tot i en
als $i \notin S$ is $a(i) = -1$.

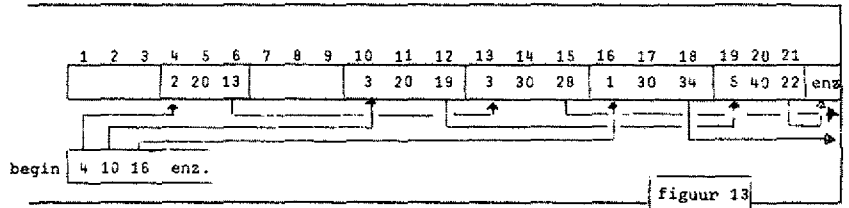
Bijv. als S bestaat uit de punten 1, 4 en 2 (zie figuur 9), dan is $as = 3$, $s(1) = 1$, $s(2) = 4$, $s(3) = 2$, $a(1) = 0$, $a(2) = 20$, $a(4) = 15$, $a(3) = a(5) = a(6) = a(7) = -1$.

In figuur 12 doorloopt ii de waarden van 1 tot as en dus $i = s(ii)$ de punten van S. Voor ieder punt i worden de met i verbonden punten j bekeken. Als $j \in S$, dus $a(j) \neq -1$, komt deze j niet in aanmerking, als $j \notin S$ wordt nagegaan of $a(i) + d(i, j)$ kleiner is dan het tot dan toe gevonden minimum m. Na afloop van het stuk programma in figuur 12 is m het gezochte minimum en jm de waarde van j waarvoor het wordt aangenomen.

3.4. Ketening

Extra moeilijkheden doen zich voor als het netwerk voortdurend aan wijzigingen onderhevig is. We willen dan niet alleen efficiënt de kortste weg bepalen, maar ook lijnstukken aan het netwerk toevoegen of uit het netwerk verwijderen. Daarvoor is de aanpak met een acces-tabel niet geschikt.

Hier kan men weer gebruik maken van ketens, een keten voor ieder punt. In een rij begin kan men dan vinden waar iedere keten begint (figuur 13).



figuur 13

Het toevoegen en verwijderen van lijnstukken gaat weer door het wijzigen van enkele links. Als men de bijkomende lijnstukken achteraan in de rij g laat aansluiten, loopt men echter de kans dat men achter in de rij vastloopt, terwijl er meer naar voren inmiddels vele lege plekken ontstaan zijn.

Daarom is het beter de lege plekken ook te ketenen. Als een lijnstuk wordt verwijderd, worden de plaatsen die het innam aan deze keten toegevoegd, heeft men nieuwe plaatsen nodig, dan neemt men ze van deze keten.

4. Slot

Het was niet de bedoeling in het voorgaande kant en klare oplossingen te bieden. Bij nadere beschouwing zal men ontdekken, dat er vele verfijningen kunnen worden aangebracht. Er is hier slechts een idee gegeven van hoe men in het geheugen structuren kan verwezenlijken.

1.3 IETS OVER SOFTWARE

1.3.1 Inleiding

Onder hardware (apparatuur) wordt verstaan de computer in de toestand, waarin deze verkeert op het moment dat hij in een krat door de fabrikant wordt opgeleverd. Het toestel is op dat moment nog tot niets in staat. Als U de stekker in het stopkontakt steekt, zullen er enige onvoorspelbare chaotische reacties ontstaan binnen de kast, en daarmee uit. Om van deze nare 'ijzerwinkel' een bruikbare machine te maken, moeten er programma's in het computergeheugen worden opgenomen die vrijwel alle 'handelingen' van het apparaat regelen.

Door de meeste vakmensen wordt onder 'software' (programmatuur) alles verstaan wat de computer laat werken, uitgezonderd de hardware zelf. Onder deze software kan men dus ook verstaan de programma's, die door (of voor) de gebruiker (ook U en uw leerlingen) zijn geschreven. Dit wordt wel de 'zachte programmatuur' (applicatie software) genoemd.

De 'harde programmatuur' (basis software) bestaat uit de programma's, die de elementaire verrichtingen van de machine en zijn bijbehorende apparaten regelen. Deze programma's worden door specialisten (al of niet in dienst van de fabrikant) geschreven. Veel van deze programma's zijn permanent in het geheugen van de computer aanwezig en vaak zodanig beveiligd dat ze niet door gebruikers kunnen worden 'beschadigd'.

Over deze harde programmatuur gaat de nu volgende beschouwing.

Om U een indruk te geven van hardeprogrammatuur nu een voorbeeld. U bent gewend om U te kunnen bedienen van de opdracht $a := \text{LEES}$. Daarbij moet het getal -851.37 in de geheugenplaats a worden opgenomen. Hoe weet de machine waar precies het getal ophoudt? Een hoe wordt dit getal, met toestandsteken en decimale punt, omgezet in de interne notatie bestaande uit alleen nullen en enen van een geheugenplaats?

Het antwoord op beide vragen is, dat dit geregeld moet worden door een programma. Aangenomen dat de LEESYM-opdracht bekend is, zodat symbool voor symbool gelezen kan worden, ziet dit programma er bijvoorbeeld als volgt uit:

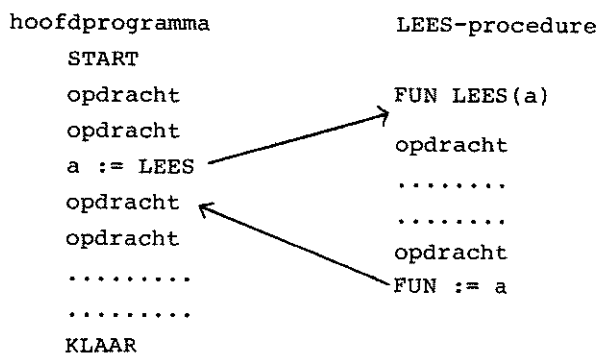
```

FUN LEES(a)
teken := LEESYM
a := 0
1  b := LEESYM
   ALS b > 9 DAN 3 ;hier wordt gecontroleerd of we
                   ;iets anders dan een cijfer te-
                   ;genkomen
2  a := 10 × a + b ;hier ziet U het 'effect' van
                   ;het tientallig stelsel
   NAAR 1
3  ALS b = 70 DAN 4 ANDERS 7 ;hier wordt gezocht
                           ;naar decimale punt
4  d := 1
5  d := d/10           ;na de decimale punt wordt het
                       ;gelezen symbool door de juiste
                       ;macht van 10 gedeeld
   c := LEESYM
   ALS c > 9 DAN 7
6  a := a + c × d
   NAAR 5
7  ALS teken = 81 DAN 8 ANDERS 9 ;negatief zijn
                               ;wordt verwerkt
8  a := -a
9  FUN := a

```

Dit programma is niet in overeenstemming met de ECOL-werkelijkheid. In hoofdstuk 3 kunt U lezen, dat er in het getal spaties mogen voorkomen. Dit programma zou het lezen van het getal beëindigen bij een spatie, want een tweede punt en elk ander niet-cijfer symbool geldt hier als stopcriterium. Als dit symbool het toestandsteken van het volgende getal is, zou dit bewaard moeten worden tot de volgende keer. Voor de duidelijkheid zien we hier maar van af.

Dat dit programma niet wordt ingesloten door START en KLAAR vindt zijn oorzaak in het feit, dat dit programma dient als onderdeel van een langer programma, zie onderstaande figuur.



Hoofdstuk 3 licht U in over de preciese werking van het FUNctie-mechanisme. Enkele aspecten die in dit voorbeeld niet gemist kunnen worden zijn:

- Men kan bij het aanroepen van de LEES-procedure een parameter meegeven, te weten: de geheugen-plaats waarin het te lezen getal terecht moet komen.
 - De in de LEES-procedure genoemde namen en regel-nummers mogen niet verward worden met eventueel identieke namen en nummers in het hoofdprogramma.
- U begrijpt dat hier nog wel enige software aan te pas komt.

Ook de opdracht LEESYM is overigens niet zonder meer duidelijk aan de machine. Zolang het gelezen symbool een cijfer is, is het probleem minder groot als bij het lezen van een letter of een leesteken. In het geheugen kunnen deze laatste nu eenmaal niet worden opgeslagen. Daarom worden de laatstgenoemde symbolen omgezet in getallen. We zullen dit de interne code noemen. Ook dit omzetten gebeurt met een programma. Dit programma kan signalen ontvangen van een ponsbandlezer of een schrapkaartlezer en deze 'omrekenen' tot interne code. Bijvoorbeeld de schrapkaart

[illegible]

veroorzaakt de volgende 'signalen':

[illegible]

Deze signalen worden omgezet in de volgende
getallen:

90 90 3 90 90 90 40 51 58 90 10 86 11
3 # # # A L S # a > b

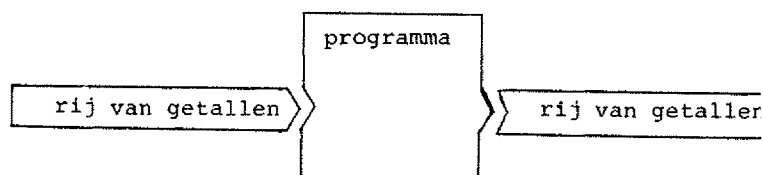
80 5 90 43 40 53 90 1 3 90 40 53 43 44 57 58 90
+ 5 # D A N # 1 3 # A N D E R S #

7 90 90 90 90 90
7 # # # # #

Deze interne code is overigens afhankelijk van de machine en zal bovendien in het algemeen in binaire code worden geschreven. Hier is de ECOL-code uit Compk § 16.1 gebruikt.

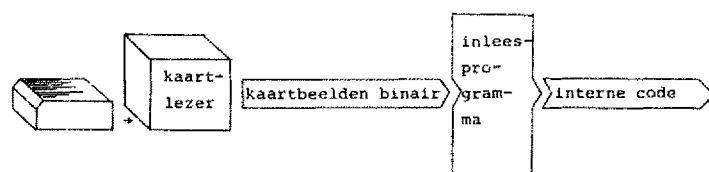
5.3.2 Schema van een programma:
maverwerking

Zoals bij het laatste voorbeeld heel duidelijk te zien is, zal een softwareprogramma vaak een rij van getallen laten omwerken tot een rij van andere getallen. Symbolisch zullen we dit als volgt aangeven:



Met opzet staat hier 'laten omwerken', omdat het in wezen niet het programma zelf is, waar de getallen ingaan en uitkomen. Het is het rekengedeelte van de computer, waarin dit gebeurt onder invloed van het programma.

Het inlezen van schrapkaarten kan nu als volgt worden geschematiseerd.

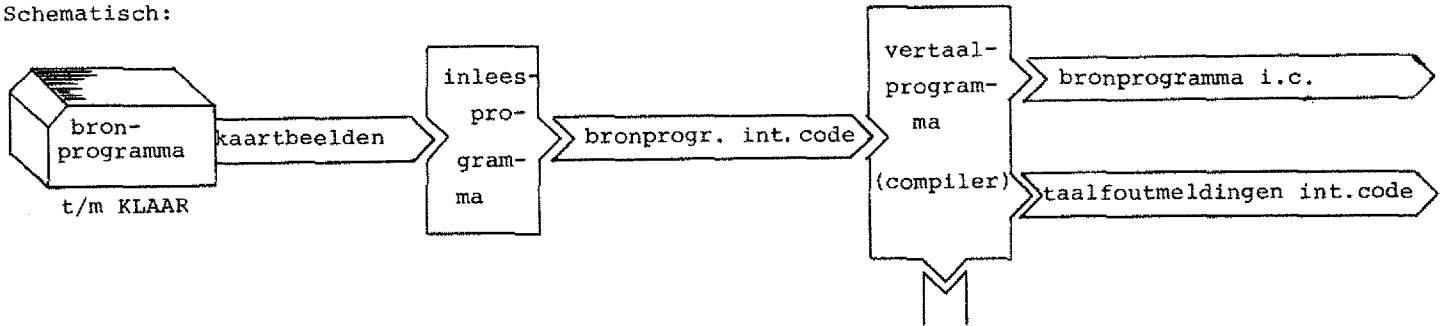


Wanneer U als gebruiker een programma indient op schrapkaarten, dan wordt dit op dezelfde wijze in interne code omgezet. Maar hiermee moet nog wel iets gebeuren, want dat 40 51 58.... enzovoort een voorwaardelijke sprongopdracht voorstelt is aan de computer niet duidelijk.

Wat wèl door een machine wordt verstaan, kunt U vinden in de paragraaf over hardware, namelijk machinecode-opdrachten van een eenvoudiger soort, binair weergegeven.

Tuw ingediende programma (het zogenaamde bronprogramma) moet worden omgezet in machinecode. Het wordt na deze behandeling objectprogramma genoemd. En dat gebeurt niet automatisch (hardware), maar weer onder invloed van een software-programma.

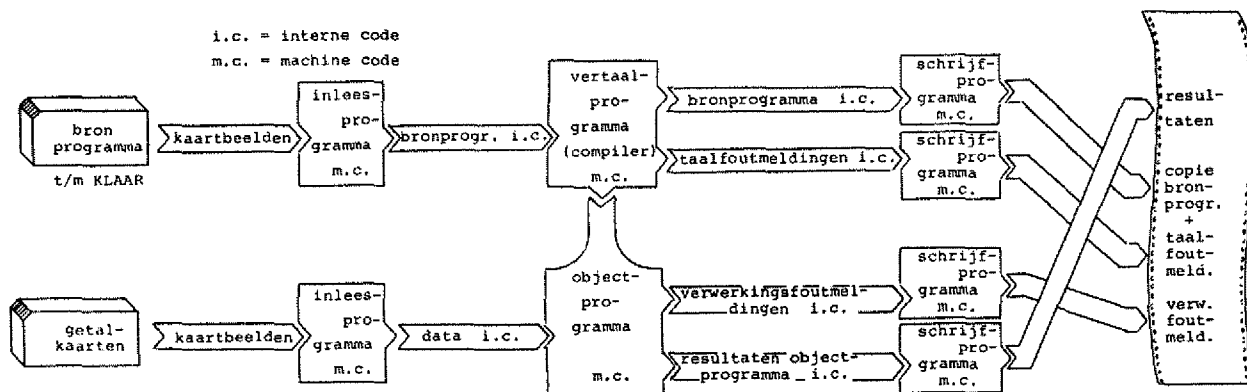
Schematisch:



Uit dit vertaalprogramma komen weer rijen getallen. Het eerst komt een herhaling van het bronprogramma. Als de interne code hiervan is omgezet in leesbaar schrift, kan de schrijver van het bronprogramma controleren of alles goed is doorgekomen. Daarna verschijnen eventueel de taalfoutmeldingen. Als een bepaalde regel van het programma luidde:

ALS 7 LEES a ANDERS

dan kan het vertaalprogramma daar met de beste wil van de wereld niets mee beginnen. Dit programma produceert dan een tekst, waaruit de fout blijkt. Dit is dan een fout tegen de ECOL-taal. Tenslotte komt er een rij binaire getallen, dat de vertaling is van het bronprogramma in het objectprogramma (machinecode). Deze rij onderscheidt zich niet van de andere programma's in machinecode (inleesprogramma, vertaalprogramma, schrijfprogramma), zodat het in het schema hieronder recht-hoekig is weergegeven. Het bijzondere van het vertaalprogramma is dus, dat één van zijn 'afschei-dingsprodukten' weer een programma (het objectprogramma) is.



Het objectprogramma in machinecode kan dan aan het werk, waarbij eventuele getallen van getalkaarten als voedsel worden toegediend. Uitvoer van het objectprogramma bestaat natuurlijk uit de resultaten, maar zo mogelijk ook uit meldingen van fouten die bij de verwerking van het programma blijken (executiefouten). Als het bronprogramma tien keer een LEES-opdracht bevat, terwijl maar zes getallen op de getalkaarten aanwezig zijn, is dat niet een fout tegen de ECOL-taal die door de vertaler wordt ontdekt, maar een fout die blijkt als het objectprogramma werkelijk draait. Een andere veel voorkomende executiefout is de eindeloze cyclus.

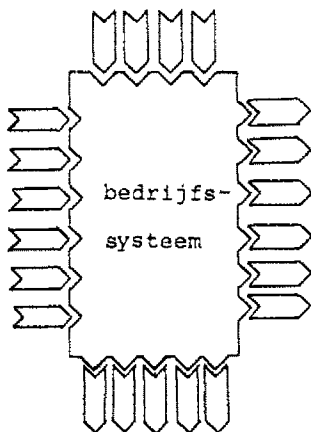
Er is een derde type fout, die niet in dit schema (figuur 1) is aangegeven: de schrapfout. Het is mogelijk dat een leerling een niet geoorloofde combinatie van hokjes aanstreept (bijvoorbeeld START en KLAAR tegelijk in kolom 4). Het inleesprogramma signaleert dit, maar geeft de foutmelding niet direct naar buiten, maar geeft in interne code een "niet bestaand symbool" door, waarna de compiler voor foutmelding zorgdraagt.

U ziet aan alle software die aan de programmaverwerking te pas komt, dat zonder deze hardware programma-tuur de computer een onhandelbaar ding is. De hardware is slechts het zichtbare fundament, waarop de software de ruime toepasbaarheid bouwt.

In de inleiding werd al gezegd: als een computerinstallatie bij U zou worden binnengebracht en U

steekt de stekker in het stopkontakt, dan is het toestel nog niet in staat iets nuttigs te doen. Er is nog geen compiler, er zijn nog geen inlees- of schrijfprogramma's. Gelukkig is er door de fabrikant wel een kleine toegang tot het computergeheugen ingebouwd, de zogenaamde 'hardware loader'. Met een speciale code kunnen hierdoor kleine programma's worden ingevoerd, als het ware met een theelepeltje. Onder de software, die als eerste naar binnen wordt gelepeld, zal men een redelijk inleesprogramma aantreffen, zodat men tenminste met de paplepel kan verdergaan.

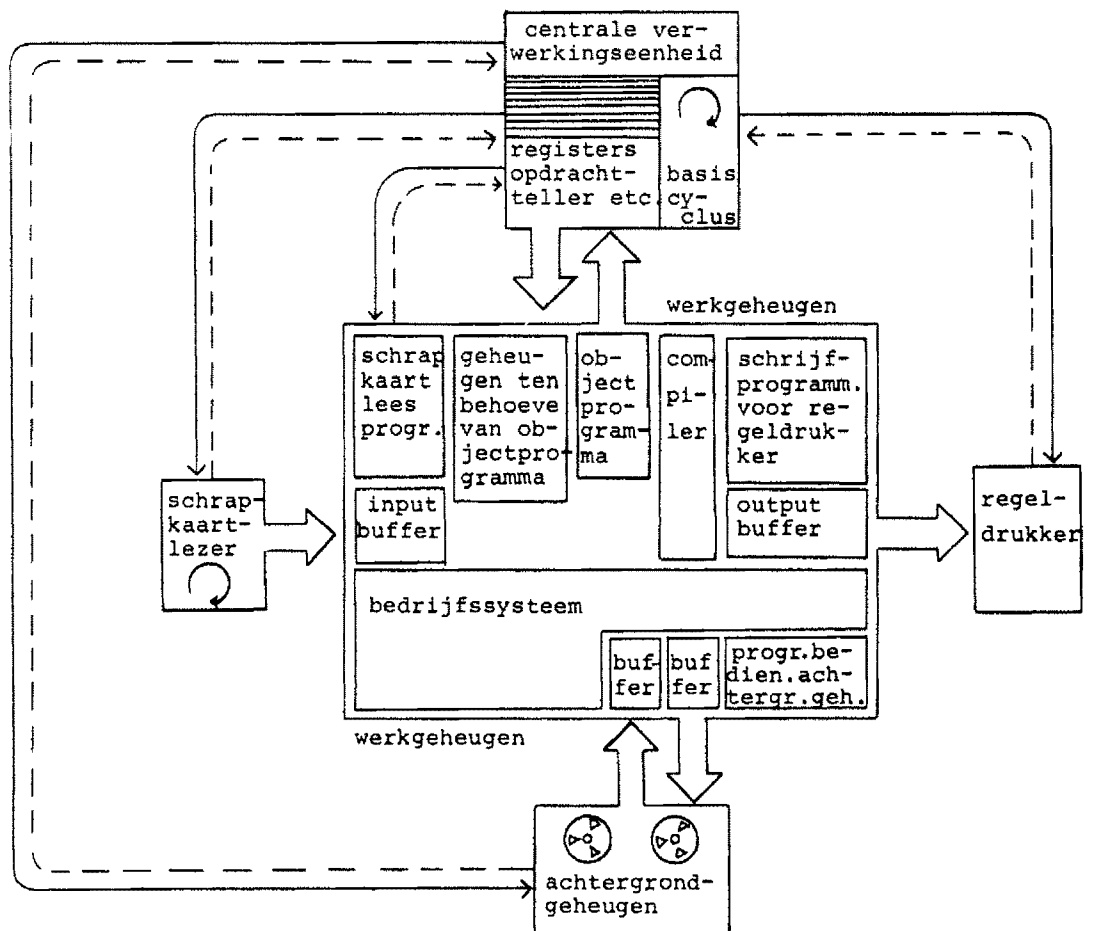
Het bovenstaande schema (figuur 1) is trouwens nog maar een gedeelte van de werkelijkheid. Een moderne centrale verwerkingseenheid van een computer, waarvoor een miljoen geheugencontacten per seconde geen uitzondering is, zal aan de hier geschetste verwerking maar een gering deel van zijn tijd besteden. De meeste tijd gaat zitten in het inlezen en schrijven. Bij moderne configuraties zijn meerdere objectprogramma's tegelijk in verwerking. 'Tegelijk' in dit geval op te vatten als: het eerste programma rekent, terwijl het tweede leest, het derde schrijft en het vierde even staat te wachten. Deze wirwar mag niet ontaarden in een chaos. De organisatie van dit geheel berust bij....., U raadt het al, een programma. Dit programma onderhoudt contact met zoveel onderdelen en zoveel andere software, dat de volgende symbolisering niet overdreven is.



Dit programma wordt ook wel bedrijfssysteem genoemd. Het samenstellen van bedrijfssystemen is een hooggekwalificeerde arbeid, waarvoor nog steeds te weinig mensen zijn opgeleid. Onder andere hieraan ontleent het Hoger Informatica Onderwijs in Nederland zijn bestaansrecht. De ontwikkeling van bedrijfssystemen is nog ver achter bij de mogelijkheden, die de hardware biedt. Voor het "bouwen" van compilers geldt overigens hetzelfde.

5.3.3 G_e_h_e_u_g_e_n_g_e_b_r_u_i_k

Figuur 1 geeft geen overzicht naar plaats maar naar tijd aan. De vier schrijfprogramma's in dit schema geven bijvoorbeeld in wezen één programma aan, op vier verschillende tijdstippen aangeroepen. In figuur 2 wordt een poging gedaan een sterk vereenvoudigde en geschematiseerde plaatsaanduiding te geven.



figuur 2

We gaan er weer vanuit, dat sprake is van één objectprogramma.

In figuur 2 geeft  een kanaal aan waardoor informatie kan stromen (de rijen getallen uit de vorige paragraaf),  een commandolijn voor de besturing vanuit de centrale verwerkingseenheid en  een verbinding voor berichten aan deze eenheid.

Het voornaamste nieuwe element is het begrip buffer. Een regeldrukker is in staat een regel van meer dan 100 symbolen in één keer af te drukken. Als het objectprogramma allemaal getallen naast elkaar laat schrijven, worden deze niet stuk voor stuk geschreven, maar verzameld tot een hele regel gevuld is. De verzamelplaats voor zo'n regel heet de output buffer. In de meeste gevallen zal het schrijfprogramma zorgen dat bij overlopen van de buffer het laatste getal niet wordt afgebroken, maar bewaard tot de volgende regel. De regel kan dan, of bij eerder ontvangen van de opdracht "nieuwe regel", in één klap worden afgedrukt. De inputbuffer speelt een soortgelijke rol in omgekeerde richting. Wanneer een schrapkaart gelezen wordt, zal deze niet schrapje voor schrapje worden doorgegeven, maar er wordt een compleet kaartbeeld in een buffer opgeslagen. Het schrapkaartleesprogramma kan nu netjes de kolommen tot superkolommen samenvoegen, detecteren op onjuiste aanstrepingen en dergelijke. Omdat het verkeer met het achtergrondgeheugen (werken met bestanden) in beide richtingen gaat, zijn hiervoor twee buffers.

Het programma voor de bediening van het achtergrondgeheugen regelt zaken als het opzoeken van het juiste stuk magneetband en het onderhouden van de buffers.

De dunne pijlen geven in het schema de commando- en berichtenlijnen aan. Als het objectprogramma een leesopdracht bevat, wordt de schrapkaartlezer volgens de getrokken lijn geactiveerd. De lezer op zijn beurt signaleert langs de gestippelde lijn het gereedstaan voor een volgende opdracht. De schrapkaartlezer houdt er een eigen (vrij langzaam) tempo op na, vandaar de eigen aanduiding van een basiscyclus.

Bij de meeste computerfiguraties moet U denken aan meer lees- en schrijfapparatuur en meer achtergrondgeheugen, de zogenaamde randapparatuur of perifere apparatuur, elk met hun eigen buffers in het werkgeheugen.

Bovendien zijn, zoals in de vorige paragraaf gezegd, meestal meerdere gebruikersprogramma's tegelijk in het geheugen opgenomen. Tenslotte zijn bij echte grote computers soms meerdere centrale verwerkingseenheden ingeschakeld. Het is overigens vrijwel nooit zo, dat deze aan eenzelfde taak (eenzelfde objectprogramma) werken. Dus in dat geval kunt U de computer denken te bestaan uit meerdere rekenmachines, die één werkgeheugen gebruiken.

Het bedrijfssysteem dat bij zo'n computer orde houdt, is van indrukwekkende omvang.

5.4 RECURSIVITEIT

Aan de hand van een aantal voorbeelden zal getracht worden het begrip recursiviteit duidelijk te maken.

Voorbeeld 1) In een syntax staat:

onder [rij kleine letters] wordt verstaan: δf [kleine letter]
 δf [rij kleine letters] [kleine letter]

Hier staat een definitie van het begrip "rij kleine letters". Rechts staat [kleine letter]. De tekst-haken geven aan, dat het om het begrip gaat en niet om de 12 letters en één spatie die tussen deze haken staan. Het begrip [kleine letter] moet elders in de syntax zijn gedefinieerd, bijvoorbeeld door opsomming van alle symbolen die men onder een [kleine letter] wenst te verstaan. De recursie bestaat nu uit het gebruik van het te definiëren begrip in de definitie. Men ziet immers aan de rechterzijde weer [rij kleine letters] voorkomen. Met [rij kleine letters] [kleine letter] wordt bedoeld een [rij kleine letters] gevolgd door een [kleine letter].

Is pbda een rij kleine letters? De vraag beantwoorden we door op te bouwen:

p is een [kleine letter] dus (volgens bovenste regel van de definitie) en [rij kleine letters]
 pb is de [rij kleine letters]p, gevolgd door de [kleine letter]b, en dus (onderste definitieregel) en [rij kleine letters]
 pbd is de [rij kleine letters]pb, gevolgd door de [kleine letter]d en dus een [rij kleine letters]
 pbda is [rij kleine letters] [kleine letter] dus [rij kleine letters].

Hieruit blijkt dat het gebruik van het te definiëren begrip in de definitie zinvol kan zijn, maar dan alleen als de constructie zodanig is, dat er iets mee kan worden 'opgebouwd' of, als U achterstevoren zou redeneren, kan worden 'afgebroken'.

Constructies als

onder [rij lettertjes] wordt verstaan: δf [lettertje]
 δf [rij lettertjes]

of nog erger

onder [rij lettertjes] wordt verstaan: [rij lettertjes]

definiëren niets.

Voorbeeld 2)

onder a^n wordt verstaan: a als $n=1$
 $n \in \{1, 2, 3, \dots\}$ $a \times a^{n-1}$ als $n > 1$

Volgens deze definitie wordt onder a^5 verstaan: $a \times a^4$ en daaronder wordt verstaan $a \times a \times a^3$ en dat is $a \times a \times a \times a^2 = a \times a \times a \times a \times a^1 = a \times a \times a \times a \times a$

U ziet hier de exponent met behulp van de definitie 'afbreken'. In voorbeeld 1 was het nut van de recursieve definitie niet erg duidelijk, omdat een omschrijving in normaal Nederlands niet veel langer zou zijn. Het laatste voorbeeld laat zien, dat een recursieve definitie niet alleen overzichtelijker, maar ook korter kan zijn dan een andere.

Voorbeeld 3) Zij $R(n, m)$ de rest bij deling van n door m , dan wordt onder

GGD(n, m) verstaan: m als $R(n, m) = 0$
 $n \in \{1, 2, 3, \dots\}$ GGD($m, R(n, m)$) als $R(n, m) > 0$
 $m \in \{1, 2, 3, \dots\}$
 $n \geq m$

Wat is GGD(231, 60) ?

$R(231, 60) = 51$ dus $\text{GGD}(231, 60) = \text{GGD}(60, 51)$
 $R(60, 51) = 9$ dus $\text{GGD}(60, 51) = \text{GGD}(51, 9)$
 $R(51, 9) = 6$ dus $\text{GGD}(51, 9) = \text{GGD}(9, 6)$
 $R(9, 6) = 3$ dus $\text{GGD}(9, 6) = \text{GGD}(6, 3)$
 $R(6, 3) = 0$ dus $\text{GGD}(6, 3) = 3$

Dit is een weergave van de algoritme van Euclides, meestal als volgt uitgevoerd:

$$\begin{array}{r}
 60/231/3 \\
 \frac{180}{51/60/1} \\
 \frac{51}{9/51/5} \\
 \frac{45}{6/9/1} \\
 + \frac{6}{3/6/2} \\
 \frac{6}{0}
 \end{array}$$

Voorbeeld 4)

onder $n!$	wordt verstaan:	1	als $n=0$
$n \in \mathbb{N}$		$n \times (n-1)!$	als $n > 0$

Dit voorbeeld zal na de vorige voorbeelden zonder meer duidelijk zijn. Dergelijke recursies kunnen bij het programmeren worden overgenomen.

```

FUN fac(n)
  ALS G(n)=n EN n>=0 DAN 1
  NR
  TEKST := "ongeeoorloofde parameter"
  a := -1
  NAAR 3
1  ALS n>0 DAN 2
  a := 1
  NAAR 3
2  a := n*fac(n-1)
3  FUN := a
  
```

} controle op het juiste karakter van n

In het programma dat hierop zou volgen kan van faculteiten van natuurlijke getallen gebruik gemaakt worden. Echter $\text{fac}(3.5) = \text{fac}(-5) = -1$ met een tekst als een soort foutmelding.

Voorbeeld 5)

De Ackermann-functie

onder $A(m,n)$	wordt verstaan:	$n+1$	als $m=0$
$m \in \mathbb{N}$		$A(m-1,1)$	als $n=0$
$n \in \mathbb{N}$		$A(m-1, A(m,n-1))$	als $m \times n > 0$

Deze zo schijnbaar eenvoudige definitie geeft al snel aanleiding tot onverkwikkelijke berekeningen, die zonder computer uiterst tijdrovend zijn. Als voorbeeld $A(2,1)$.

$A(2,1) =$	$A(1,3) =$
$A(1, A(2,0)) =$	$A(0, A(1,2)) =$
$A(1, A(1,1)) =$	$A(0, A(0, A(1,1))) =$
$A(1, A(0, A(1,0))) =$	$A(0, A(0, A(0, A(1,0)))) =$
$A(1, A(0, A(0,1))) =$	$A(0, A(0, A(0, A(0,1)))) =$
$A(1, A(0,2)) =$	$A(0, A(0, A(0,2))) =$
$A(1,3) =$	$A(0, A(0,3)) =$
	$A(0,4) = 5$

U ziet dat bij het uitwerken van de definitie de problemen zich opstapelen, totdat een vorm bereikt wordt waarvan de waarde bekend is ($A(0,1)=2$). Daarna verlagen we de stapel weer tot we vastlopen bij $A(1,3)$. De problemen stapelen zich weer op tot een waarde die zonder meer bekend is. Daarna kunnen we terugkeren naar de bodem van de stapel. Dit werken met een stapel is een veelvoorkomend principe bij computers. In de meeste gevallen zijn zelfs hardware voorzieningen voor het werken met een stapel ingebouwd (zie 4.6 in § 5.1).

Voorbeeld 6)

Het partitieprobleem.

Onder een partitie van een positief geheel getal verstaan we dit getal zelf of een wijze om dit getal

te schrijven als som van een aantal positieve gehele getallen. Zo is $2+2+1$ een partitie van 5. Alle verschillende partities van 5 zijn: 5, $4+1$, $3+2$, $3+1+1$, $2+2+1$, $2+1+1+1$, $1+1+1+1+1$.

U ziet hieruit dat $4+1$ en $1+4$ niet als verschillende partities zullen worden aangemerkt. Partities schrijven we voortaan in niet stijgende termen. Een partitie van 5 kan dus beginnen met een 5, met een 4, met een 3, met een 2 of met een 1. Een partitie van 5 beginnend met een 2 heeft de gedaante $2 + [\text{partitie van } (5-2)]$. Maar niet elke [partitie van 3] is hierin toegestaan, want $2+3$ is tegen de zojuist gemaakte afspraak. Daarom is het volgende wél juist.

Een partitie van 5 beginnend met 2 heeft de gedaante $2 + [\text{partitie van } (5-2) \text{ met termen } \leq 2]$.

In het algemeen heeft een partitie van a de gedaante:

$$n + [\text{partitie van } (a-n) \text{ met termen } \leq n] \\ n \leq a, \quad a \text{ en } n \text{ positief geheel.}$$

De probleemstelling luidt nu: bepaal het aantal partities van een positief geheel getal a . Voor de berekening hiervan voeren we de grootte $P(m,n)$ in, zijnde het aantal partities van m met termen $\leq n$. Het aantal partities van 5, dat is $P(5,5)$ kan bepaald worden uit:

$$P(5,5) = 1 + P(1,4) + P(2,3) + P(3,2) + P(4,1).$$

De eerste 1 geeft de triviale partitie van 5 aan, namelijk 5 zelf. $P(4,1)$ is het aantal partities van 4 met termen ≤ 1 , dus uitsluitend enen. Er is maar één zo'n partitie, namelijk $1+1+1+1$.

In het algemeen is $P(m,1) = 1$. Ook $P(1,n) = 1$, want het getal 1 heeft slechts de triviale partitie. Verder volgt uit de definitie dat bijvoorbeeld $P(3,3) = P(3,4) = P(3,5)$ enz.

Terwijl $P(3,3) = P(3,2) + \text{de triviale partitie}$.

In het algemeen: $P(m,n) = P(m,m-1) + 1$ als $m \leq n$.

Nu nog het geval dat $m > n > 1$. De recursieve relatie van $P(m,n)$ moet bestaan met P -functies waarvan de m of de n lager is. Bijvoorbeeld:

$$P(5,3) \\ \overbrace{3+2, 3+1+1, 2+2+1, 2+1+1+1, 1+1+1+1+1}^{P(5,2)}$$

$P(5,3)$ kan verknipt worden in iets $+ P(5,3-1)$. Dat iets is het aantal partities van 5, die met een 3 beginnen, dat is het aantal partities van $5-3$, met termen kleiner dan 3, ofwel $P(5-3,3)$.

Kortom $P(5,3) = P(5-3,3) + P(5,3-1)$.

In het algemeen $P(m,n) = P(m-n,n) + P(m,n-1)$ als $m > n > 1$.

Het aantal partities $P(a,a)$ van het positieve gehele getal a kan berekend worden met de recurente betrekking:

$P(m,n) =$	1	als m of $n = 1$
	$P(m,m-1)+1$	als $m \leq n$
	$P(m-n,n)+P(m,n-1)$	als $m > n > 1$

Een programma hiervoor, waarin de waarden van m en n niet in de FUN-deklaratie worden gecontroleerd is het volgende:

```

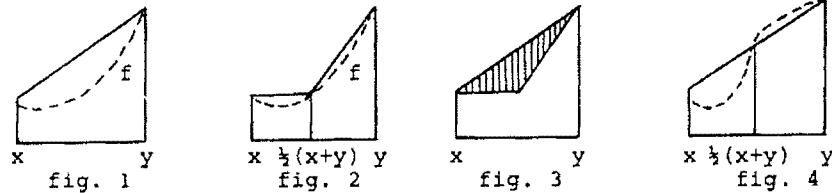
FUN p(m,n)
ALS m=1 OF n=1 DAN 1 ANDERS 2
1  b := 1
   NAAR 4
2  ALS m>n DAN 3
   b := p(m,m-1) + 1
   NAAR 4
3  b := p(m-n,n)+p(m,n-1)
4  FUN := b
   START
   MET 1, a:=1, 20
   NR
   TEKST := "P("
   SCHRIJF(2,0) := a
   SCHRIJF SYM := 71
   SCHRIJF(2,0) := a
   TEKST := ")"
   SCHRIJF(6,0) := P(a,a)
   HERHAAL
   KLAAR

```

Uitwerkingen:

P(1 , 1) =	1
P(2 , 2) =	2
P(3 , 3) =	3
P(4 , 4) =	5
P(5 , 5) =	7
P(6 , 6) =	11
P(7 , 7) =	15
P(8 , 8) =	22
P(9 , 9) =	30
P(10 , 10) =	42
P(11 , 11) =	56
P(12 , 12) =	77
P(13 , 13) =	101
P(14 , 14) =	135
P(15 , 15) =	176
P(16 , 16) =	231
P(17 , 17) =	297
P(18 , 18) =	385
P(19 , 19) =	496
P(20 , 20) =	627

Voorbeeld 7) De trapeziumregel.



De oppervlakte $\frac{1}{2}(y-x)(f(x)+f(y))$ van het trapezium is een benadering voor $\int_x^y f(t)dt$ (fig. 1). We zullen in het vervolg deze oppervlakte aangeven met $T(x,y)$.

In het algemeen zal $T(x, \frac{1}{2}(x+y)) + T(\frac{1}{2}(x+y), y)$ een betere benadering geven (fig. 2). Een criterium voor de precisie kan zijn: $V(x,y) = T(x,y) - T(x, \frac{1}{2}(x+y)) - T(\frac{1}{2}(x+y), y)$ (fig. 3).

Op dit laatste zijn wel uitzonderingen te bedenken, zoals in fig. 4, waarin $V(x,y)=0$, zonder dat er van een goede benadering sprake is. We zonderen dit soort gevallen uit door te eisen dat $f''(t) \neq 0$ op (x,y) , zodat de functie f op dit interval uitsluitend convex of uitsluitend concaaf is.

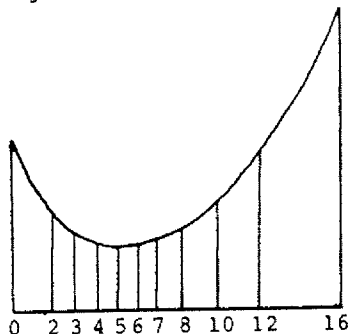
De benadering $I(a,b)$ van $\int_a^b f(t)dt$, met $f''(t) \neq 0$ op (a,b) kan berekend worden met een tolerantie ϵ met de volgende recursie:

$$\begin{array}{ll} I(x,y) = & T(x, \frac{1}{2}(x+y)) + T(\frac{1}{2}(x+y), y) \quad \text{als } |V(x,y)| < \epsilon \frac{y-x}{b-a} \\ \text{anders} & I(x, \frac{1}{2}(x+y)) + I(\frac{1}{2}(x+y), y) \end{array}$$

Hoe kleiner het interval $[x,y]$, hoe kleiner de "fout" $V(x,y)$ die we mogen toestaan. De som van alle foutjes op de intervallen mag immers ϵ niet te boven gaan, zodat per interval slechts het $\frac{y-x}{b-a}$ - deel van ϵ getolereerd kan worden.

De recursieve toepassing van de trapeziumregel heeft het voordeel, dat niet meer gerekend wordt dan strikt noodzakelijk. Dit wil zeggen, er wordt alleen tot halvering van een interval overgegaan, als het nauwkeurigheidscriterium op dat interval niet gehaald wordt.

fig. 5



Stel $I(0,16)$ van een zeker functie moet bepaald worden met een tolerantie 0,32.

Men schrijft een programma dat werkt volgens de aangegeven recursie (fig. 5).

$$V(0,16) > 0,32 \Rightarrow I(0,16) = I(0,8) + I(8,16).$$

De computer legt $I(8,16)$ even apart en gaat verder met $I(0,8)$.

$$V(0,8) \geq 0,16 \Rightarrow I(0,8) = I(0,4) + I(4,8) \quad I(4,8) \text{ apart leggen, bovenop } I(8,16)$$

$$V(0,4) \geq 0,8 \Rightarrow I(0,4) = I(0,2) + I(2,4)$$

$$V(0,2) < 0,4 \Rightarrow I(0,2) \text{ wordt berekend. Er ligt nu een stapel te wachten van de inhoud: } \rightarrow \begin{array}{l} I(2,4) \\ I(4,8) \\ I(8,16) \end{array}$$

De bovenste van de stapel komt nu het eerst aan bod (het principe LIFO, Last In - First Out)

$$V(2,4) \geq 0,4 \Rightarrow I(2,4) = I(2,3) + I(3,4)$$

$$V(2,3) < 0,2 \Rightarrow I(2,3) \text{ berekend}$$

$$V(3,4) < 0,2 \Rightarrow I(3,4) \text{ berekend}$$

$I(2,4)$ berekend

$I(0,4)$ berekend

$I(2,4)$
 $I(4,8)$
 $I(8,16)$

$I(3,4)$
 $I(4,8)$
 $I(8,16)$

$$V(4,8) \geq 0,8 \Rightarrow I(4,8) = I(4,6) + I(6,8)$$

$$V(4,6) \geq 0,4 \Rightarrow I(4,6) = I(4,5) + I(5,6)$$

enz.

$I(6,8)$
 $I(8,16)$

$I(5,6)$
 $I(6,8)$
 $I(8,16)$

U ziet hierbij weer het principe van de stapel.

Slotopmerking:

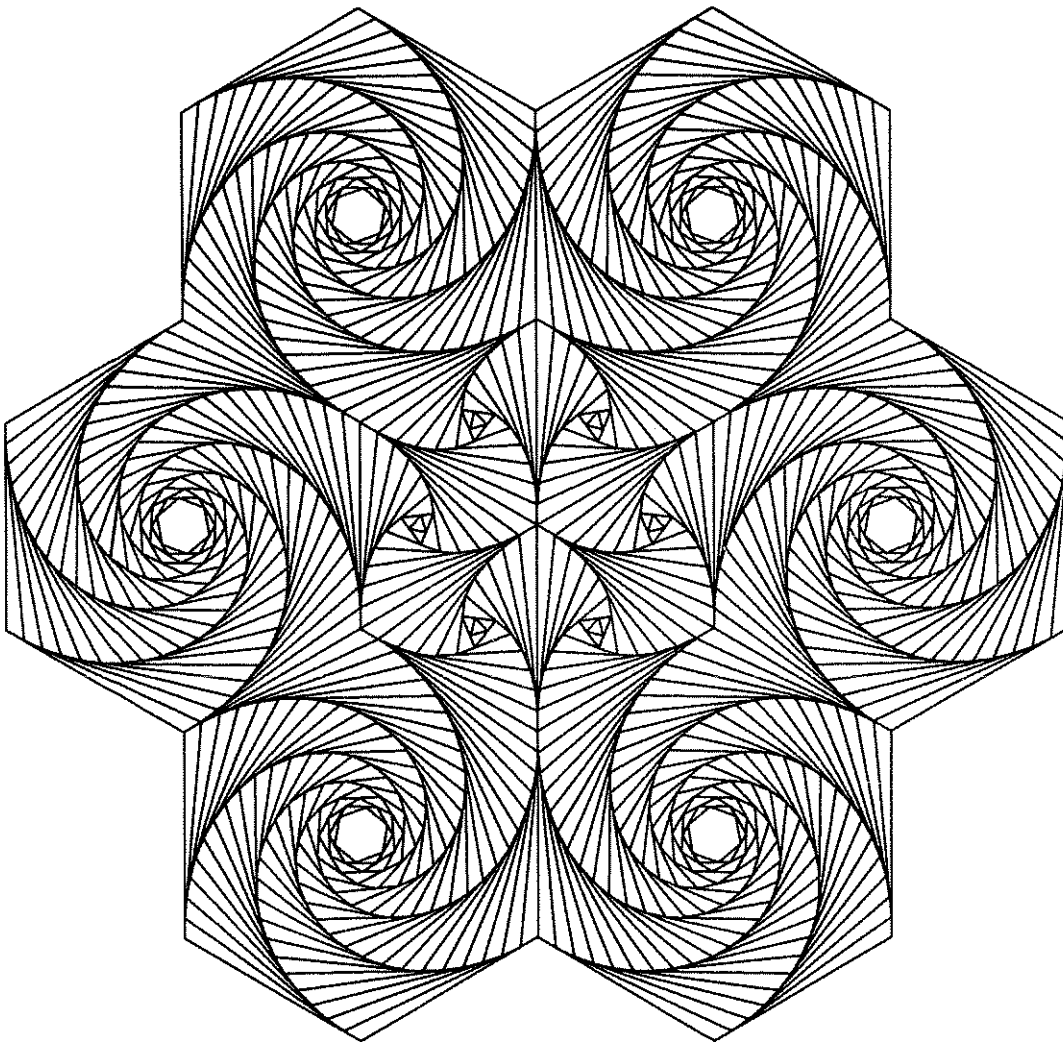
Enkele van de gegeven voorbeelden hebben een minder uitgesproken recursief karakter. Ze zijn ook te definiëren (of te berekenen) door een herhaling van stappen, die omschreven kunnen worden in termen van reeds gedefinieerde begrippen (iteratie).

In voorbeeld 3 kan de definitie luiden: deel het kleinste op het grootste getal. Als de rest niet nul is, neem dan de deler als deeltal en de rest als deler en voer deze deling uit. Herhaal de laatste regel, totdat de rest nul is. De GGD is gelijk aan de laatste deler. Zo kan ook de berekening van $n!$ als een iteratief proces worden aangegeven:

```
prod := 1
m := 1
1  prod := prod × m
   ALS m=n DAN 2
   m := m+1
   NAAR 1
2  nfac := prod
```

Tenslotte kan ook de integraal van voorbeeld 7 volgens een iteratie benaderd worden, maar dan door telkenmale halvering van alle intervallen. $I(a,b) = I(a,b,n)$ als $|I(a,b,n) - I(a,b,n-1)| < \epsilon$ waarbij voor

$$\Delta t = \frac{b-a}{2^n}, \quad I(a,b) = T(a, a + \Delta t) + T(a + \Delta t, a + 2\Delta t) + \dots T(b - \Delta t, b)$$



6 Programmering

6.1 DE ECOL-SCHRAPKAART

Vervolg			START	?																															
0	0	0	TEKST	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BE	0	BESTAND
1	1	1	NR	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DA	1	DAN
2	2	2	SPATIE	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	AN	2	ANDERS
3	3	3	SCHRIJF	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LE	3	LEES
4	4	4	SCHRIJM	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.
5	5	5	SCHRIJSYM	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=	5	=
6	6	6	ALS	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+	6	+
7	7	7	NAAR	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-	7	-
8	8	8	RIJHERNAAL	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X	8	X
9	9	9	KLAAR	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/

De schrapkaart is U bekend van de volkstelling, maar dan met andere opdruk. Het gebruik is als volgt:

- 1) Eén regel van het programma wordt op één of meer kaarten aangegeven
- 2) De regel wordt geschreven aan de bovenzijde van de kaart, één symbool per hokje (afgezien van regelnummers en hoofdletterwoorden)
- 3) De regel wordt daarna gecodeerd door potloodstreepjes in de aangegeven hokjes
- 4) De kolommen 1, 2 en 3 dienen voor het aangeven van een regelnummer
- 5) Kolom 4 voor de aangegeven hoofdletterwoorden, die alle aan het begin van een opdracht staan
- 6) Kolom 5 evenzo
- 7) Kolom 6 en 7, verder alle volgende tweetallen kolommen tot en met 34 en 35, beschouwen als telkens één superkolom voor het aangeven voor één symbool of hoofdletterwoord
- 8) Met één streepje (in het desbetreffende hokje) worden aangegeven:

!	0	1	2	3	4	5	6	7	8	9
?	BESTAND	DAN	ANDERS	LEES	.	:=	+	-	x	/
- 9) Een symbool dat linksonder in het hokje staat wordt aangeduid door een streep in dat hokje plus een streep in het hokje links boven in de superkolom
- 10) Analooq voor een symbool rechtsonder. Met SY wordt zowel SYM als LEESYM bedoeld
- 11) Is één kaart onvoldoende om een opdracht te bevatten dan mag worden doorgeschreven op de volgende(n), mits deze wordt(worden) gemarkeerd door een streep in het bovenste hokje van kolom 3
- 12) Hoofdletters geeft men aan als de overeenkomstige kleine letter plus een streep in hokje met de omhooggerichte pijl (in totaal drie streepjes in die superkolom)
- 13) De kaarten mogen na beschrijving niet van volgorde verwisseld worden
- 14) Aantekeningen van de gebruikers op de achterzijde van de kaart.

Hieronder volgen een aantal voorbeelden van het gebruik van de schrapkaart.

		x := 8.3 / m																					
		Vervolg START																					
0	0	0	TEKST	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	1	NR	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	2	2	SPATIE	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
3	3	3	SCHRIJF	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
4	4	4	SCHRIJF	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5
5	5	5	SCHRIJF	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
6	6	6	ALS	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7
7	7	7	NAAR	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
8	8	8	RUIJ	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9
9	9	9	KLAAR	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10

Het wordt-teken staat midden in een hokje en wordt daarom met één streepje aangegeven.

Elke letter wordt er onder met twee streepjes aangegeven.

Hieronder is de opdracht x := 8.3/m op de kaart aangestreept. De punt kunt U vinden in het hokje naast de 4.

De opdracht TEKST := "Jan"

Let vooral op de aanhalingstekens. Die zijn te vinden linksonder (dus twee streepjes) in het hokje naast de 3.

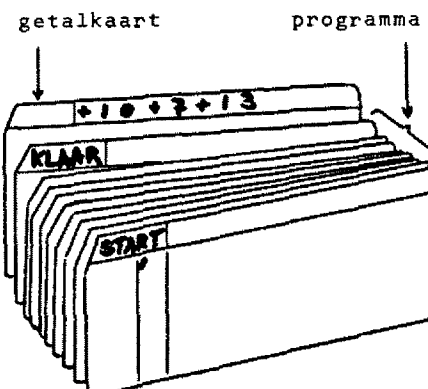
Het is ook mogelijk hoofdletters op de schrapkaart aan te strepen. U doet hetzelfde als met een kleine letter, dat is dus twee streepjes. Maar bovendien moet er een streepje in het hokje met de naar boven gerichte pijl. Dus in het totaal drie streepjes. Hieronder ziet U de hoofdletter J.

		TEKST := "Jan"																					
		Vervolg START																					
0	0	0	TEKST	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	1	NR	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	2	2	SPATIE	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
3	3	3	SCHRIJF	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
4	4	4	SCHRIJF	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5
5	5	5	SCHRIJF	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
6	6	6	ALS	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7
7	7	7	NAAR	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
8	8	8	RUIJ	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9
9	9	9	KLAAR	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10

		ALS a > 6 DAN 6 ANDERS 8																					
		Vervolg START																					
0	0	0	TEKST	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	1	NR	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	2	2	SPATIE	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
3	3	3	SCHRIJF	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
4	4	4	SCHRIJF	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5
5	5	5	SCHRIJF	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
6	6	6	ALS	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7
7	7	7	NAAR	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
8	8	8	RUIJ	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9
9	9	9	KLAAR	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10

We zien hoe de opdracht ALS .. DAN .. ANDERS .. op de schrapkaart moet worden aangestreept. Elk van deze drie woorden kan worden aangegeven met één streepje, voor DAN in het hokje met DA en voor ANDERS in het hokje met AN. Bovendien wordt getoond hoe een regelnummer voor de opdracht wordt aangegeven. Op de schrapkaart kan LEES worden aangegeven door één streepje in het hokje met LE. Vergeet U bij deze opdracht niet de getalkaart(en) achter het programma.

		62 SCHRIJF := a																					
		Vervolg START																					
0	0	0	TEKST	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	1	NR	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	2	2	SPATIE	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
3	3	3	SCHRIJF	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
4	4	4	SCHRIJF	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5
5	5	5	SCHRIJF	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
6	6	6	ALS	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7	6	7
7	7	7	NAAR	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
8	8	8	RUIJ	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9
9	9	9	KLAAR	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10



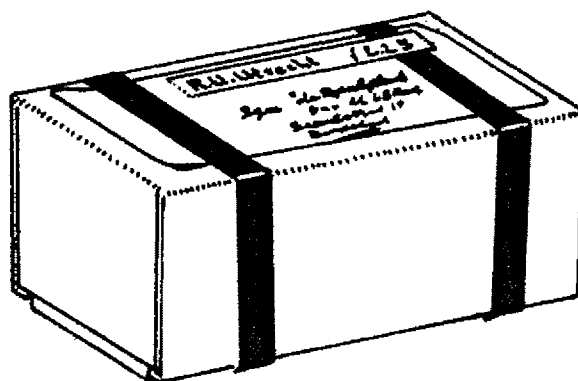
		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100	
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100			
1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43																																																																																																																					

6.3 HET VERZENDEN VAN SCHRAPKAARTEN

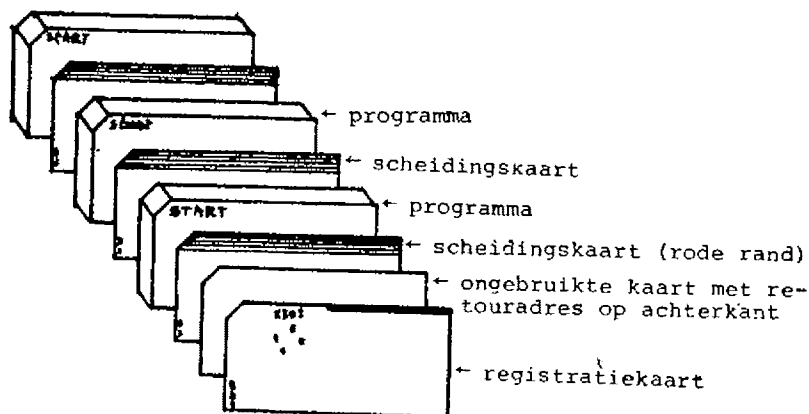
Voor het insturen van schrapkaarten voor de programmaverwerking stelt het I.O.W.O. standaardverpakking ter beschikking.

In elk op te sturen doosje moeten de voorste twee kaarten zorg dragen voor een correcte retourzending:

1. De registratiekaart, een met een nummer geponste kaart, met een gedeeltelijk zwarte bovenrand. Elke docent heeft een eigen viercijferig nummer. Deze kaart zorgt ervoor, dat de papieren die uit de regeldrukker komen (de "output") automatisch van een retouradres worden voorzien.
2. Een gewone schrapkaart, waarvan aan de voorkant niets geschreven mag zijn. Op de achterkant vermeldt U het adres waar U de schrapkaarten terug wenst te ontvangen (zie figuur). Na de verwerking wordt deze kaart gebruikt als etiket voor het verzendoosje.



Standaardverpakking bij retourzending



Behalve door de twee genoemde kaarten moet elk programma afzonderlijk worden voorafgegaan door een scheidingskaart. Dit is een kaart met een rode bovenrand.

↑
 ruimte voor postzegels
 ↓

← ruimte voor
plakband →

Sgm "de Speelplaats"

t.a.v. dhr. L.E. Raar

Schoolstraat 15

Dorpsstad

← ruimte voor
plakband →

6.4 KOSTEN PROGRAMMAVERWERKING

Bij het volgende wordt een schatting gegeven van de officiële kosten per programma. Let wel: niet het bedrag dat de school volgend jaar in rekening wordt gebracht, maar een berekening van de totale kosten, waarvan het Rijk een groot gedeelte voor haar rekening kan nemen. De genoemde bedragen blijven voor verantwoording van de schrijver.

We gaan uit van 10 programma's per leerling per jaar. Dit is een gemiddelde. Sommige leerlingen schrijven meer. Werken in groepsverband drukt het gemiddelde weer. Een aantal programma's wordt door schrapfouten meermalen aangeboden enz. Behalve bij punt 7 nemen we 50 leerlingen per school, die computerkunde volgen.

Kosten per programma

1. Programma's worden centraal verponst (zoals in 1973 nog voor scholen gebeurt bij het CRI te Leiden). Voordelen:
 - a) eenvoudigste werkwijze voor de leerlingen
 - b) geen investeringskosten voor de school
 Nadelen:
 - d) 'turn around'-tijd vaak meer dan 1 week
 - e) leerlingen hebben geen enkel contact met apparatuur
 - f) onleesbaar handschrift geeft fouten
 - g) lange programma's of getalbanden zijn minder welkom
 - h) niet herstelde ponsfouten
 - i) toepassing op grote schaal onmogelijk door personeelstekort
 - j) verbetering van een programmafout kan alleen door programma te herhalen.

2 sec computertijd (à f 600,-/uur)	f 0,30
10 min ponswerk (à f 12,-/uur)	" 2,--
porti heen en terug	" 0,10
	f 2,40

2. Schrapkaart (zie ook het artikel hierover)
 Voordelen:
 - a) lengte en inhoud van het programma en lengte van de getalband zijn vrij
 - b) programma is eenvoudig te verbeteren (kaarten vervangen)
 - c) verkeerd schrijven van hoofdletterwoorden komt niet voor (staan kant en klaar op de kaart)
 - d) geen investeringskosten voor de school

- e) 'turn around'-tijd minder dan 1 week
- Nadelen:
- f) aanstrepen programma kost 5 à 10 min
 - g) kaarten mogen niet door elkaar raken
 - h) zwaar in het versturen
 - i) geen contact met apparatuur

2 sec computertijd	f 0,30
30 kaarten	" 0,30
porti	" 0,20
	f 0,80

3. Telexverbinding met computercentrum (1 toestel per 50 leerlingen)

Voordelen:

- a) geen andere fouten dan door de leerling zelf gemaakt
 - b) leerling heeft enig contact met apparatuur, zij het nauwelijks verband houdend met computers
 - c) 'turn around'-tijd van 1 dag
- Nadelen:
- d) investering van f 2500,- per toestel per jaar
 - e) leerling moet ervaring krijgen in machineschrijven
 - f) intypen van een programma kost 20 min
 - g) intypen moet in de meeste gevallen buiten de lesuren

2 sec computertijd	f 0,30
huur en aanleg telextoestel	
(f 2500,-/500)	" 5,--
huur en aanleg ontvangtoestel op rekencentrum (f 2500,-/5000)	" 0,50
telefoonkosten	" 2,40
	f 8,20

4. Ponsapparaat op school, ponsband opsturen.

Voordelen:

- a) geen andere fouten dan door de leerling zelf gemaakt
 - b) zie 3.b
 - c) 'turn around'-tijd van 4 dagen
- Nadelen:
- d) investering van f 8000,- per toestel zie verder 3.e, 3.f en 3.g.

2 sec computertijd	f 0,30
afschrijving, renteverlies, onder- houd (f 1600,-/500)	" 3,20
porti	" 0,05
	<u>f 3,55</u>

Op dit bedrag kan bezuinigd worden in het geval dat het apparaat bediend wordt door (niet) onderwijzend personeel en bovendien het aantal leerlingen, dat met dit ene apparaat bediend wordt groot is. Bij 500 leerlingen zakt de prijs naar f 2,40 per programma.

. De Terminal (indirecte verbinding met de computer = on line)

Voordelen:

- a) fouten worden onmiddellijk gecorrigeerd
- b) leerling heeft contact met apparatuur
- c) onmiddellijke verwerking

Nadelen: zie 3e, 3f en 3g.

2 sec computertijd	f 0,30
huur en aanleg terminal (f 3500,-/500)	" 7,--
telefoonkosten	" 4,80
	<u>f 12,10</u>

Op de telefoonkosten kan worden bespaard als in ieder telefoondistrict een time-sharing systeem beschikbaar is.

. De Terminal (off line)

Voordelen:

- a) geen andere fouten dan door de leerling zelf gemaakt
- b) leerling heeft contact met apparatuur
- c) 'turn around'-tijd van 1 dag

Nadelen: zie 3e, 3f en 3g.

2 sec computertijd	f 0,30
huur en aanleg terminal (f 3500,-/500)	" 7,--
telefoonkosten	" 1,--
	<u>f 8,30</u>

. Computer op school (van minstens 100 leerlingen computerkunde)

Voordelen:

- a) direct contact met de computer
- b) zie verder bij 5 en 6

Nadelen: zie 3e, 3f, 3g en hoge investering van f 150.000,-.

ponskosten of terminalgebruik bij 1000 programma's/jaar	f 1,60.
renteverlies, afschrijving en onder- houd computer (f 15.000,-/1000)	" 15,--
	<u>f 16,60</u>

Het gebruik van formulierenlezers wordt nog onderzocht. Zo'n lezer werkt in principe hetzelfde als de optische lezer voor schrapkaarten. De formulieren (velletjes papier iets groter dan de schrapkaart) zijn lichter dan de schrapkaart en daarom goedkoper en gemakkelijker te versturen.

Het nadeel is dat de lezer iets beperkter is dan de schrapkaartlezer. Bij de schrapkaart kan gebruik gemaakt worden van 12 hokjes per kolom, waarvan er overigens maar 11 in gebruik zijn vanwege de schrijfruimte die bovenaan de kaart is vrijgehouden.

Bij een formulier zijn er 10 hokjes per kolom. Hierdoor moet worden overgegaan op een andere codering, wat op zichzelf niet zo'n bezwaar is, als er maar een aanvaardbare indeling kan worden gevonden.

De kosten zijn nog onbekend, maar liggen waarschijnlijk lager dan die van een schrapkaartenprogramma.

N.B. De voorgaande cijfers zijn gebaseerd op de situatie in 1971. Sommige prijzen van apparatuur zijn inmiddels verlaagd, arbeidsloon en posttarieven zijn niet onaanzienlijk gestegen.

6.5 LIJST VAN BESTANDEN

BESTAND- NUMMER	OPGAVE UIT COMEK	AANTAL GP'S	OMSCHRIJVING
10	13.7 17.2	1000	333 overschrijvingen: nraf bedr nrbij, afgesloten door -1
11	§13	1000	1000 saldi
12 t/m 14	personeels- administratie	1000	onbezet
15		200	geb, laatste twee cijfers van de geboortejaren van 200 werknemers
16		200	bs, burgerlijke staat van 200 werknemers: -2, -1, +1, +2
17		200	ak, aantallen kinderen van 200 werknemers
18		200	jid, laatste twee cijfers van jaar van indiensttreding
19		200	sal, salarissen van 200 werknemers
20		150	belastingtabel van Compk II bladzijde 27
21		300	geb, geboortedatum gecodeerd
22		300	overl, datum van overlijden gecodeerd
23		300	mv, man (1) of vrouw (-1)
24	bevolkingsregister	300	va, nummer van de vader
25		300	egen, nummer van de echtgenoot of echtgenote
26		300	oudk, nummer van het oudste kind
27		300	brzus, nummer van broer of zuster die in leeftijd volgt
28 t/m 30		1000	onbezet
31 t/m 40	§15	20	elk 20 getallen ≤ 100: aantal vrije vliegtuigplaatsen
41 t/m 50	§15	40	plaatsbezetting genoteerd in enen en nullen
51	blz. 44 en	300	100 woorden van drie letters, niet alfabetisch
	17.4		N.B. In Compk II, blz. 44 is dit aangegeven met BESTAND(80)
52	16.5	1000	tekst
53	16.7	1000	100 woorden van 10 of minder letters
54	16.8	1000	tekst in geheimschrift, verschuiving over n, n is eerste getal
55	16.11	1000	tekst zonder overgangen op nieuwe regel
56	16.17	1000	Tom Poes verhaal met Professor Prwytzkofsky
57	16.18	1000	tekst met opmerkingen tussen haakjes
58	16.20	500	geldbedragen in oude Engelse notatie
59 t/m 63	§16	500	teksten in ECOL
64	17.1	251	getallen, voorafgegaan door hun aantal
65	17.5	500	50 woorden van 20 of minder letters gescheiden door spaties
66	17.7	63	63 telefoonnummers in opklimmende volgorde
67	17.7	1000	63 bijbehorende abonnees
68	17.8	630	63 woorden ≤ 10 letters in alfabetische volgorde

Op de volgende bladzijden vindt U de gehele of gedeeltelijke inhoud van een groot aantal bestanden.

RESTAND 10

2	0.51	24
908	106.12	754
361	298.60	455
517	498.98	607
293	23.92	648
28	527.95	725
609	18.36	534
310	107.13	139
968	26.04	658
56	41.68	554
749	63.31	81
743	18.92	354
431	145.01	716
593	632.54	600
25	8.60	282
55	193.67	648
815	127.56	990
546	20.36	198
478	101.10	523
590	27.73	146
311	18.30	208
266	59.75	316
155	145.87	216
615	98.99	236
246	97.83	305
436	43.07	183
58	399.74	376
562	204.93	989
907	42.00	836
718	45.71	743
847	143.45	951
146	56.84	296
131	54.21	473
140	8.48	210
296	251.54	903
927	114.52	754
615	56.32	356
233	10.27	477
493	27.94	28
33	56.76	314
19	161.96	403
930	22.34	834
795	33.39	540
315	158.80	666
760	79.54	457
201	36.55	38
534	76.35	492
397	478.54	384
124	21.58	551
442	93.26	667
460	140.74	396
425	11.11	812

76	141.12	864
622	69.95	429
956	89.80	957
595	19.49	718
338	52.68	420
167	150.61	171
984	55.96	749
415	25.90	638
226	10.04	543
686	72.41	641
792	73.57	995
718	30.17	105
720	29.65	60
957	31.69	14
365	36.36	553
431	80.85	449
297	171.60	252
873	69.32	147
625	6.21	742
90	24.31	490
5	56.90	564
526	24.99	600
392	135.78	929
104	358.60	910
295	297.97	50
252	287.20	402
80	9.78	845
779	24.91	399
596	16.96	578
941	182.25	566
154	0.94	674
45	39.00	544
646	145.74	798
179	17.35	351
332	15.39	872
52	16.82	471
570	105.77	790
135	246.19	283
537	144.44	754
353	231.08	235
703	39.21	629
203	159.99	916
602	205.70	815
962	10.78	957
176	110.79	439
395	38.35	358
723	80.35	810
112	306.25	724
237	276.16	492
480	36.42	515
659	88.18	585
764	149.32	219
334	345.06	807
872	68.64	131
691	37.21	652
886	20.05	117
941	115.82	650
277	70.07	537
318	110.51	518
455	48.79	224

137	174.39	726
75	364.38	176
716	201.87	763
230	27.34	371
931	251.36	135
468	1.21	862
939	48.82	869
265	51.95	51
348	76.12	72
369	191.90	802
871	57.96	797
177	26.51	809
246	138.14	284
40	153.44	355
932	153.79	317
160	162.26	387
903	229.85	270
483	37.97	553
528	14.82	83
748	25.22	611
344	80.66	234
585	92.59	364
254	155.90	462
338	19.55	32
412	138.21	790
3	12.93	707
850	61.65	117
447	128.16	317
606	64.94	416
807	86.99	222
904	52.71	326
734	40.38	390
653	54.64	652
874	11.28	777
302	159.71	70
763	22.02	325
836	266.66	60
22	106.40	744
434	34.56	852
518	147.85	979
78	31.30	920
902	116.32	8
145	266.44	235
669	83.97	393
760	9.15	690
647	9.73	733
439	25.62	408
590	22.99	923
310	18.01	206
252	44.25	851
934	75.53	775
123	122.55	140
523	49.34	632
715	456.23	474
63	102.19	279
90	4.50	459
645	574.55	163
947	20.90	612
26	197.54	728
566	278.29	542

195	4.40	503
376	146.45	238
495	11.02	176
888	232.92	430
550	17.48	13
371	40.12	651
69	31.35	1000
425	78.12	436
268	73.47	719
363	114.54	778
472	176.79	729
91	10.43	788
166	8.24	991
765	71.60	182
512	121.84	622
610	28.81	18
152	6.11	996
558	36.89	833
778	18.15	943
537	16.76	99
800	115.27	904
734	77.03	623
100	22.08	295
17	137.75	342
682	17.80	843
413	5.79	627
744	11.21	942
303	34.61	33
439	258.15	598
737	167.15	127
374	47.04	892
25	18.96	823
616	10.63	67
511	108.16	368
747	152.23	973
198	283.08	872
236	30.63	464
595	134.33	81
172	138.94	950
996	35.89	736
131	52.31	271
43	116.35	746
711	338.87	406
264	26.14	10
18	20.17	943
989	54.13	606
131	26.15	216
781	97.60	714
327	48.25	358
302	2.58	438
687	85.09	980
280	67.96	443
782	112.86	23
959	296.30	60
180	94.31	48
211	117.37	249
726	89.92	24
199	90.67	604
953	16.98	359
258	145.66	284

201	141.28	735
403	2.64	531
51	8.15	18
603	146.88	193
773	46.18	264
747	190.53	392
316	52.86	626
944	34.42	292
115	298.79	662
580	55.59	346
768	124.19	358
259	144.41	258
333	37.13	870
573	30.51	428
805	111.41	798
321	0.88	174
41	160.05	430
606	69.78	561
156	379.30	473
963	3.09	514
193	150.02	930
412	181.66	327
581	53.75	274
102	6.79	489
661	44.70	215
960	115.52	473
331	26.93	440
488	211.30	892
563	58.82	331
607	93.46	187
348	142.40	433
237	37.24	744
337	123.81	236
977	411.94	109
213	27.00	508
85	7.20	681
574	84.11	252
613	270.78	91
858	320.83	39
419	159.41	57
596	57.34	263
355	66.54	727
14	36.51	547
994	39.89	31
774	294.99	873
492	0.46	605
412	246.12	742
541	32.78	812
640	15.60	110
646	194.65	329
743	63.48	798
939	169.24	449
555	322.71	871
417	41.26	282
356	176.80	741
281	125.70	766
844	309.67	139
767	133.80	144
710	78.43	878
627	106.88	303

Het restant van bestand 10 staat op bladzijde 110.

97	35.34	919
170	32.73	156
582	215.51	74
519	146.20	944
252	0.37	757
491	142.32	137
17	33.42	711
250	55.42	320
924	53.85	135
640	69.52	247
31	50.56	118
875	35.34	939
83	236.44	923
551	1.14	507
65	296.63	85
983	106.39	87
378	3.92	836
331	70.36	61
188	108.46	295
206	215.60	437
223	58.01	637
142	05.89	190
52	0.95	600
488	39.05	543
620	105.50	337
843	12.93	85
569	33.37	409
102	33.76	607
736	113.41	449
412	71.27	353
472	0.16	768
411	45.71	512
236	0.58	945
417	0.69	292
313	69.26	186
387	405.18	416
354	193.30	955
975	59.01	899
617	158.09	212
874	19.10	140
730	95.81	291
-1		

Hierboven staat het restant van bestand 10.

In bestand 11 hieronder zijn de rekeningnummers en de streepjes in werkelijkheid niet opgenomen. Ze staan hier uitsluitend voor de duidelijkheid.

BESTAND 11

1	305.80	2	48.07	3	133.51	4	215.45	5	354.30
6	78.64	7	142.99	8	314.83	9	350.13	10	5.72
11	349.50	12	362.70	13	273.58	14	30.60	15	10.36
16	771.91	17	49.59	18	444.18	19	1518.28	20	26.83
21	235.43	22	1027.33	23	107.62	24	198.77	25	143.50
26	133.22	27	166.97	28	71.42	29	655.45	30	438.66
31	154.66	32	331.35	33	135.25	34	490.38	35	703.20
36	120.55	37	631.06	38	420.59	39	256.62	40	435.33
41	79.14	42	379.35	43	280.57	44	680.00	45	50.30
46	2.04	47	253.26	48	219.16	49	426.94	50	455.88
51	199.03	52	20.12	53	84.88	54	832.35	55	51.06
56	292.18	57	583.55	58	88.78	59	415.51	60	65.58
61	23.25	62	461.35	63	288.55	64	77.16	65	24.71
66	198.84	67	143.80	68	131.00	69	179.29	70	37.09
71	87.64	72	170.50	73	38.00	74	145.53	75	292.56
76	487.89	77	255.98	78	118.37	79	631.08	80	479.57
81	111.87	82	911.39	83	199.78	84	216.57	85	232.41
86	44.39	87	116.74	88	528.00	89	42.12	90	431.14
91	84.96	92	322.47	93	528.63	94	32.30	95	809.36
96	199.29	97	355.17	98	189.44	99	873.82	100	35.74
101	506.23	102	31.29	103	21.99	104	345.18	105	88.86
106	107.17	107	663.43	108	922.11	109	91.71	110	542.28
111	285.46	112	473.01	113	376.89	114	251.87	115	50.08
116	1003.06	117	528.47	118	128.52	119	182.23	120	42.60
121	34.40	122	51.78	123	23.03	124	41.78	125	242.87
126	208.13	127	1.08	128	193.69	129	36.21	130	11.30
131	421.87	132	44.44	133	143.21	134	198.15	135	344.83
136	239.68	137	239.70	138	412.22	139	258.74	140	633.13
141	297.31	142	114.73	143	170.72	144	230.49	145	629.61
146	722.24	147	309.67	148	885.09	149	220.12	150	129.97
151	71.27	152	251.07	153	356.52	154	388.14	155	147.18
156	45.81	157	324.29	158	429.49	159	280.19	160	77.97
161	84.17	162	191.72	163	1804.24	164	974.12	165	239.48
166	15.59	167	412.44	168	19.05	169	711.99	170	1.57
171	493.82	172	34.89	173	23.88	174	241.43	175	1561.55
176	17.80	177	365.00	178	89.88	179	35.83	180	1554.70
181	6.18	182	790.27	183	85.18	184	53.34	185	480.01
186	410.46	187	126.66	188	76.37	189	240.83	190	328.67
191	11.87	192	449.21	193	1499.54	194	17.02	195	394.59
196	36.28	197	395.20	198	261.68	199	752.89	200	86.25
201	79.36	202	132.27	203	439.42	204	58.78	205	3.85
206	116.70	207	909.79	208	147.71	209	31.85	210	679.49
211	378.06	212	303.23	213	399.51	214	588.97	215	74.04
216	907.47	217	117.05	218	207.18	219	31.65	220	724.69
221	302.15	222	1606.53	223	113.75	224	48.45	225	92.41
226	20.83	227	71.29	228	40.84	229	121.32	230	587.05
231	577.68	232	47.51	233	170.36	234	25.71	235	280.73
236	285.94	237	556.88	238	156.96	239	981.77	240	247.06
241	1100.61	242	32.62	243	283.62	244	155.24	245	22.71
246	784.98	247	457.34	248	996.24	249	58.21	250	93.15
251	580.08	252	393.53	253	113.71	254	227.27	255	1088.34
256	142.77	257	1024.52	258	467.02	259	246.85	260	228.12

261	-	425.10	262	-	325.42	263	-	70.67	264	-	457.76	265	-	161.16
266	-	186.66	267	-	196.19	268	-	902.72	269	-	1116.06	270	-	267.05
271	-	343.33	272	-	279.21	273	-	78.92	274	-	52.56	275	-	924.15
276	-	530.42	277	-	172.56	278	-	727.56	279	-	270.51	280	-	78.97
281	-	92.08	282	-	120.71	283	-	301.38	284	-	72.97	285	-	0.11
286	-	16.87	287	-	331.37	288	-	205.65	289	-	1.60	290	-	209.59
291	-	823.49	292	-	591.64	293	-	159.93	294	-	445.09	295	-	308.15
296	-	63.09	297	-	26.88	298	-	455.76	299	-	234.29	300	-	379.51
301	-	506.34	302	-	314.62	303	-	436.48	304	-	242.15	305	-	450.75
306	-	129.45	307	-	692.00	308	-	175.79	309	-	80.66	310	-	270.88
311	-	111.54	312	-	461.67	313	-	203.83	314	-	1083.34	315	-	386.48
316	-	683.89	317	-	30.16	318	-	127.82	319	-	17.75	320	-	228.31
321	-	101.03	322	-	201.42	323	-	203.52	324	-	309.43	325	-	367.07
326	-	1088.93	327	-	493.12	328	-	152.38	329	-	129.13	330	-	114.38
331	-	898.64	332	-	177.74	333	-	438.18	334	-	138.26	335	-	619.78
336	-	172.27	337	-	213.63	338	-	920.80	339	-	280.00	340	-	450.84
341	-	621.90	342	-	424.94	343	-	265.80	344	-	320.96	345	-	439.62
346	-	192.36	347	-	81.85	348	-	110.12	349	-	2.29	350	-	62.94
351	-	477.08	352	-	220.15	353	-	376.82	354	-	8.90	355	-	147.07
356	-	159.19	357	-	12.76	358	-	13.54	359	-	134.08	360	-	174.14
361	-	36.34	362	-	130.23	363	-	590.13	364	-	332.33	365	-	82.53
366	-	226.23	367	-	321.88	368	-	101.03	369	-	135.10	370	-	86.79
371	-	78.17	372	-	135.19	373	-	426.85	374	-	59.99	375	-	25.64
376	-	629.64	377	-	1913.89	378	-	36.18	379	-	121.77	380	-	1116.90
381	-	46.10	382	-	135.23	383	-	233.55	384	-	201.26	385	-	66.74
386	-	155.59	387	-	449.99	388	-	358.10	389	-	526.23	390	-	436.23
391	-	960.13	392	-	127.84	393	-	100.61	394	-	55.12	395	-	167.97
396	-	746.82	397	-	305.78	398	-	728.46	399	-	450.06	400	-	42.56
401	-	72.29	402	-	223.15	403	-	775.09	404	-	393.39	405	-	105.43
406	-	312.61	407	-	179.05	408	-	195.01	409	-	27.38	410	-	57.72
411	-	54.11	412	-	171.13	413	-	710.67	414	-	314.12	415	-	887.16
416	-	190.73	417	-	310.59	418	-	680.59	419	-	509.20	420	-	9.17
421	-	189.30	422	-	763.44	423	-	90.13	424	-	48.52	425	-	438.92
426	-	13.90	427	-	501.44	428	-	382.19	429	-	138.63	430	-	105.81
431	-	652.29	432	-	2.17	433	-	1173.31	434	-	17.07	435	-	565.92
436	-	159.31	437	-	669.17	438	-	828.37	439	-	127.62	440	-	201.96
441	-	854.13	442	-	838.12	443	-	660.19	444	-	72.67	445	-	367.05
446	-	529.83	447	-	336.05	448	-	205.79	449	-	949.05	450	-	462.72
451	-	326.91	452	-	857.58	453	-	139.04	454	-	103.06	455	-	748.25
456	-	632.79	457	-	191.05	458	-	92.67	459	-	54.55	460	-	283.00
461	-	548.79	462	-	188.80	463	-	308.54	464	-	858.90	465	-	250.68
466	-	39.79	467	-	51.33	468	-	836.36	469	-	922.56	470	-	533.44
471	-	168.29	472	-	867.38	473	-	189.83	474	-	347.74	475	-	325.93
476	-	565.81	477	-	880.34	478	-	345.62	479	-	140.69	480	-	273.21
481	-	13.58	482	-	131.40	483	-	192.55	484	-	1012.67	485	-	854.58
486	-	319.56	487	-	266.95	488	-	250.61	489	-	106.50	490	-	35.12
491	-	312.99	492	-	38.17	493	-	181.37	494	-	67.90	495	-	362.74
496	-	1030.24	497	-	635.07	498	-	3.83	499	-	2.24	500	-	551.87
501	-	652.64	502	-	35.40	503	-	131.20	504	-	618.95	505	-	256.10
506	-	359.77	507	-	307.51	508	-	601.98	509	-	10.93	510	-	1609.58
511	-	367.30	512	-	491.68	513	-	499.34	514	-	644.78	515	-	154.78
516	-	298.92	517	-	265.81	518	-	477.70	519	-	803.31	520	-	83.67
521	-	51.64	522	-	591.37	523	-	159.61	524	-	451.45	525	-	289.42
526	-	100.63	527	-	381.76	528	-	36.94	529	-	524.38	530	-	0.61
531	-	242.97	532	-	316.02	533	-	30.50	534	-	245.46	535	-	450.74
536	-	109.81	537	-	53.75	538	-	89.14	539	-	1195.60	540	-	479.95
541	-	142.86	542	-	235.70	543	-	117.95	544	-	49.39	545	-	53.50
546	-	318.08	547	-	281.00	548	-	189.83	549	-	375.60	550	-	205.33
551	-	274.66	552	-	859.38	553	-	931.39	554	-	479.45	555	-	288.06
556	-	37.44	557	-	405.77	558	-	205.31	559	-	130.95	560	-	162.15
561	-	111.95	562	-	140.24	563	-	940.60	564	-	691.10	565	-	1403.77
566	-	928.59	567	-	131.62	568	-	97.09	569	-	46.92	570	-	308.94
571	-	548.73	572	-	58.48	573	-	144.55	574	-	35.58	575	-	656.07
576	-	329.88	577	-	2.54	578	-	878.67	579	-	84.30	580	-	3.42
581	-	389.91	582	-	228.94	583	-	86.62	584	-	352.44	585	-	272.04
586	-	67.98	587	-	162.24	588	-	331.56	589	-	77.42	590	-	286.83
591	-	90.02	592	-	703.17	593	-	83.35	594	-	157.35	595	-	137.63
596	-	37.99	597	-	1104.92	598	-	25.16	599	-	408.37	600	-	504.72
601	-	164.80	602	-	44.35	603	-	139.34	604	-	229.03	605	-	180.71
606	-	195.92	607	-	15.46	608	-	144.68	609	-	87.18	610	-	13.31
611	-	190.46	612	-	467.28	613	-	10.35	614	-	353.62	615	-	233.47
616	-	222.85	617	-	1313.79	618	-	132.51	619	-	1056.44	620	-	995.53
621	-	198.07	622	-	166.70	623	-	29.20	624	-	279.89	625	-	223.65
626	-	147.04	627	-	873.63	628	-	547.88	629	-	182.27	630	-	378.83
631	-	274.30	632	-	301.93	633	-	702.70	634	-	6.62	635	-	397.23
636	-	153.15	637	-	1623.38	638	-	167.88	639	-	189.81	640	-	120.11
641	-	484.12	642	-	76.65	643	-	173.63	644	-	122.04	645	-	12.22
646	-	418.89	647	-	40.39	648	-	191.52	649	-	10.76	650	-	234.19

651	-	169.49	652	-	228.68	653	-	739.86	654	-	370.27	655	-	214.32
656	-	243.91	657	-	20.65	658	-	269.39	659	-	560.93	660	-	268.58
661	-	546.79	662	-	311.59	663	-	258.87	664	-	378.82	665	-	204.82
666	-	258.38	667	-	25.98	668	-	109.56	669	-	162.24	670	-	404.25
671	-	482.53	672	-	233.08	673	-	246.79	674	-	446.17	675	-	111.08
676	-	58.52	677	-	49.32	678	-	205.26	679	-	325.90	680	-	249.88
681	-	311.32	682	-	11.77	683	-	947.42	684	-	718.72	685	-	794.38
686	-	84.21	687	-	55.44	688	-	461.03	689	-	443.27	690	-	109.08
691	-	76.39	692	-	714.96	693	-	226.70	694	-	313.64	695	-	123.04
696	-	47.03	697	-	32.10	698	-	26.75	699	-	256.54	700	-	459.23
701	-	40.17	702	-	65.25	703	-	360.55	704	-	13.70	705	-	172.89
706	-	1054.30	707	-	80.91	708	-	568.19	709	-	447.86	710	-	363.83
711	-	493.20	712	-	541.64	713	-	387.58	714	-	41.11	715	-	365.54
716	-	291.51	717	-	805.76	718	-	506.16	719	-	385.87	720	-	117.11
721	-	238.00	722	-	491.01	723	-	81.85	724	-	115.24	725	-	830.79
726	-	211.34	727	-	226.90	728	-	92.64	729	-	298.55	730	-	526.14
731	-	136.55	732	-	135.81	733	-	223.05	734	-	257.32	735	-	520.79
736	-	622.65	737	-	124.20	738	-	30.08	739	-	141.71	740	-	642.25
741	-	114.44	742	-	50.57	743	-	72.61	744	-	113.16	745	-	59.54
746	-	28.93	747	-	528.66	748	-	67.93	749	-	119.81	750	-	67.27
751	-	534.47	752	-	67.88	753	-	107.14	754	-	141.25	755	-	13.50
756	-	132.20	757	-	188.00	758	-	2006.94	759	-	509.22	760	-	35.01
761	-	1315.92	762	-	26.11	763	-	283.92	764	-	259.24	765	-	6.58
766	-	325.92	767	-	530.34	768	-	18.91	769	-	55.87	770	-	169.06
771	-	651.26	772	-	493.83	773	-	60.82	774	-	301.71	775	-	235.11
776	-	9.87	777	-	743.99	778	-	71.22	779	-	199.54	780	-	68.77
781	-	225.03	782	-	60.88	783	-	48.33	784	-	181.62	785	-	824.89
786	-	105.49	787	-	477.99	788	-	240.41	789	-	221.16	790	-	719.49
791	-	571.85	792	-	371.10	793	-	24.82	794	-	43.84	795	-	190.27
796	-	953.62	797	-	6.98	798	-	81.08	799	-	139.64	800	-	302.84
801	-	544.22	802	-	89.38	803	-	761.53	804	-	50.55	805	-	454.99
806	-	762.13	807	-	178.29	808	-	6.44	809	-	550.23	810	-	437.62
811	-	536.85	812	-	29.95	813	-	812.68	814	-	236.02	815	-	120.79
816	-	34.74	817	-	134.39	818	-	568.16	819	-	317.74	820	-	171.23
821	-	214.88	822	-	5.45	823	-	102.92	824	-	30.18	825	-	501.55
826	-	92.76	827	-	30.03	828	-	10.59	829	-	818.53	830	-	24.97
831	-	34.50	832	-	297.70	833	-	84.92	834	-	583.90	835	-	304.82
836	-	183.34	837	-	216.81	838	-	478.72	839	-	264.17	840	-	112.24
841	-	586.06	842	-	12.76	843	-	7.92	844	-	189.78	845	-	914.91
846	-	20.19	847	-	10.94	848	-	66.70	849	-	197.14	850	-	113.89
851	-	440.59	852	-	231.99	853	-	551.92	854	-	33.81	855	-	468.54
856	-	106.71	857	-	28.93	858	-	421.49	859	-	310.63	860	-	124.09
861	-	53.39	862	-	1083.30	863	-	356.20	864	-	22.88	865	-	109.04
866	-	214.44	867	-	64.36	868	-	84.33	869	-	470.78	870	-	13.40
871	-	283.86	872	-	325.79	873	-	627.35	874	-	113.54	875	-	75.07
876	-	542.36	877	-	1435.23	878	-	224.08	879	-	750.87	880	-	422.47
881	-	68.47	882	-	1129.53	883	-	108.09	884	-	149.93	885	-	747.84
886	-	938.17	887	-	2.15	888	-	130.06	889	-	523.09	890	-	644.11
891	-	96.45	892	-	154.20	893	-	36.40	894	-	358.58	895	-	464.50
896	-	2161.56	897	-	749.86	898	-	207.60	899	-	407.12	900	-	15.29
901	-	5.60	902	-	174.86	903	-	60.74	904	-	776.58	905	-	484.81
906	-	609.86	907	-	256.79	908	-	390.43	909	-	183.01	910	-	378.54
911	-	302.14	912	-	496.36	913	-	171.01	914	-	36.18	915	-	157.18
916	-	226.89	917	-	47.38	918	-	128.30	919	-	272.87	920	-	100.39
921	-	750.06	922	-	810.27	923	-	45.77	924	-	422.32	925	-	64.95
926	-	10.48	927	-	12.93	928	-	172.89	929	-	1455.99	930	-	33.14
931	-	175.40	932	-	154.47	933	-	275.56	934	-	444.34	935	-	938.43
936	-	111.00	937	-	232.08	938	-	1255.26	939	-	83.75	940	-	337.76
941	-	420.71	942	-	150.95	943	-	553.14	944	-	148.67	945	-	24.77
946	-	2.25	947	-	83.06	948	-	197.11	949	-	814.07	950	-	19.61
951	-	73.39	952	-	41.22	953	-	97.14	954	-	61.34	955	-	153.70
956	-	798.85	957	-	498.93	958	-	434.53	959	-	36.84	960	-	163.93
961	-	170.72	962	-	280.53	963	-	179.20	964	-	539.68	965	-	871.60
966	-	510.52	967	-	265.13	968	-	38.57	969	-	1596.08	970	-	862.09
971	-	144.86	972	-	70.93	973	-	102.67	974	-	136.71	975	-	65.29
976	-	224.14	977	-	13.68	978	-	842.38	979	-	972.46	980	-	450.97
981	-	347.89	982	-	644.50	983	-	221.77	984	-	739.48	985	-	494.17
986	-	760.17	987	-	70.47	988	-	183.73	989	-	137.61	990	-	333.38
991	-	327.03	992	-	1164.46	993	-	48.54	994	-	102.44	995	-	757.26
996	-	629.18	997	-	183.67	998	-	132.95	999	-	394.22	1000	-	148.43

Persoonsnummer	BESTANDEN				
	15	16	17	18 en	19
1	39	2	4	58	1966.00
2	36	2	2	53	1526.00
3	11	2	1	54	3214.00
4	41-	1	0	65	1541.00
5	39	2	3	57	2146.00
	47	2	0	66	1267.00
	23-	1	0	57	2040.00
	44	2	2	60	1571.00
	47	2	3	65	1055.00
10	43	1	0	59	1547.00
	45	2	3	61	1327.00
	27-	1	0	61	2546.00
	41-	1	0	67	1694.00
	49-	2	1	66	1661.00
15	44	2	1	61	1327.00
	19	2	4	43	1811.00
	43	1	3	66	2192.00
	38	2	6	55	1451.00
	56	1	0	72	1033.00
20	36	2	3	61	1999.00
	23	2	0	45	2632.00
	37	2	3	61	2171.00
	43	1	0	68	1679.00
	9	2	1	39	2299.00
25	9	2	1	37	2765.00
	45	2	3	66	1531.00
	44	2	1	67	2033.00
	11	2	1	52	3663.00
	33-	1	0	47	1392.00
30	38	2	4	59	1513.00
	22	2	2	46	2148.00
	38	2	4	65	2211.00
	49-	2	2	67	1331.00
	40-	2	4	64	1376.00
35	53-	1	0	73	1629.00
	27-	2	1	55	2693.00
	49-	2	1	67	1442.00
	55	1	0	72	999.00
	20	2	2	43	2725.00
40	35	2	3	64	2495.00
	29-	1	0	61	2555.00
	47	2	2	55	1416.00
	43	1	0	65	1624.00
	49-	2	1	69	1125.00
45	54-	1	0	73	1032.00
	9	2	1	44	3331.00
	42-	1	0	64	1622.00
	45	2	3	64	1623.00
	16	2	3	56	3036.00
50	42-	1	0	62	1393.00
	51-	1	0	63	1343.00
	15	2	1	42	1974.00
	41-	1	1	61	1577.00
	41-	1	1	66	1963.00
55	41-	1	0	63	9109.00
	43	2	4	61	1326.00
	13	2	3	59	1414.00
	50-	1	0	66	1299.00
	40-	1	0	62	1283.00
60	44	2	2	63	1517.00
	43	1	1	63	1166.00
	42-	1	0	65	1352.00
	12	2	0	57	2355.00
	17	2	1	53	2219.00
65	32-	1	0	56	1450.00
	43	2	4	60	1253.00
	43	2	3	66	1194.00
	49-	2	1	67	1820.00
	31-	1	0	54	1715.00

70	22	2	1	41	1927.00
	13	2	1	54	3552.00
	56	1	0	72	1096.00
	42	1	0	67	1697.00
	55	1	0	72	1115.00
75	41-	1	0	60	1945.00
	40-	2	5	62	1926.00
	36	2	3	55	2427.00
	25	2	2	61	2603.00
	40	2	2	59	1946.00
80	37	2	2	55	1273.00
	39	2	3	55	1725.00
	45	2	3	67	1137.00
	47	2	2	69	1747.00
	44	2	4	63	1620.00
85	55	1	0	71	395.00
	44	2	0	62	1834.00
	39	2	3	57	2343.00
	34	1	0	52	1496.00
	43	2	1	67	1817.00
90	11	2	0	33	2566.00
	42-	2	2	56	1534.00
	35	2	4	53	2431.00
	15	2	3	39	3173.00
	45	2	2	62	1243.00
95	54-	1	0	71	1463.00
	45	2	5	65	1332.00
	44	2	4	65	1923.00
	43	2	1	66	1392.00
	45	2	3	67	1431.00
100	40	2	4	59	2044.00
	43	1	0	63	2137.00
	32-	1	0	55	1596.00
	52-	1	0	69	953.00
	43	1	0	63	2046.00
105	11	2	1	31	2306.00
	20	2	2	37	2241.00
	39	2	3	65	2237.00
	25	2	3	50	1711.00
	46	2	6	64	1671.00
110	34	1	0	54	2137.00
	37	2	4	65	2339.00
	15	2	3	53	2649.00
	50-	1	0	66	1085.00
	51-	1	0	67	993.00
115	44	2	0	64	2107.00
	35	2	3	54	2562.00
	36	2	3	60	2299.00
	49-	2	1	63	1567.00
	36	2	2	62	1507.00
120	56	1	0	72	1247.00
	29-	1	3	57	2291.00
	27-	2	1	53	1933.00
	33	2	4	62	1362.00
	3	2	1	36	2631.00
125	39	2	2	59	1735.00
	51-	1	0	63	1043.00
	49-	2	1	66	1331.00
	37	2	3	59	1521.00
	17	2	1	51	2636.00
130	56	1	0	72	356.00
	43	1	3	61	1734.00
	24	2	2	43	1397.00
	16	2	3	43	2945.00
	33	2	6	64	1396.00
135	43	2	2	64	1269.00
	34	1	3	55	1763.00
	38	2	5	56	2375.00
	32-	1	0	54	1824.00
	36	2	2	53	1443.00

BESTAND 31	4	5	18	22	11	16	30	17	54	39	40	61	68	38	63	27	65	75	93	100
BESTAND 32	6	7	12	9	20	11	44	44	47	59	34	46	31	54	69	43	63	100	56	87
BESTAND 33	0	2	15	13	19	23	48	40	29	35	36	27	79	72	50	40	63	45	65	45
BESTAND 34	3	3	13	10	13	32	28	14	32	43	63	66	54	92	69	70	84	98	82	65
BESTAND 35	7	7	8	10	15	21	38	32	37	38	43	36	51	48	98	67	79	100	100	65
BESTAND 36	2	0	9	14	22	24	26	52	59	30	44	62	59	58	67	51	70	91	60	100
BESTAND 37	0	4	12	13	18	26	41	35	35	43	58	49	57	54	81	67	84	57	94	100
BESTAND 38	0	3	13	18	12	25	31	31	37	56	68	25	27	52	77	94	62	75	83	100
BESTAND 39	3	10	18	5	7	21	35	29	24	51	22	53	45	82	30	27	85	78	100	47
BESTAND 40	5	10	8	7	16	18	38	53	56	54	40	55	24	51	34	69	56	100	69	88

140	43	2	1	55	1543.20
141	25	2	3	49	2273.20
142	37	2	3	62	1746.00
143	34	1	3	56	1451.20
144	45	2	3	67	2041.55
145	33	1	3	52	2523.00
146	15	2	3	45	1352.00
147	42	1	3	57	1341.35
148	36	1	3	72	1133.00
149	46	2	6	66	1567.00
150	10	2	1	56	3372.00
151	32	1	3	54	1923.00
152	43	1	3	66	1134.20
153	43	1	3	66	1441.38
154	35	2	1	52	1633.20
155	35	2	3	55	2256.20
156	43	1	3	61	1971.20
157	50	1	3	63	1131.20
158	14	2	2	53	2031.00
159	46	2	4	67	1246.00
160	14	2	1	36	3853.00
161	40	2	2	64	1454.00
162	23	2	2	42	1655.00
163	41	1	3	66	2066.20
164	43	2	3	64	1641.20
165	37	2	3	64	1655.20
166	42	1	3	59	1244.00
167	45	2	3	66	1543.00
168	55	1	3	71	1442.00
169	43	2	3	65	1624.00
170	35	2	4	63	1574.00
171	33	1	3	53	1415.00
172	40	2	2	57	1235.00
173	41	1	3	62	1334.00
174	43	1	3	63	1752.00
175	23	1	3	47	1731.00
176	36	2	2	61	1577.00
177	44	2	2	62	1714.00
178	29	1	3	64	1453.20
179	40	2	4	63	1631.00
180	43	1	3	67	2150.00
181	26	2	3	57	1321.00
182	25	2	3	47	2646.00
183	27	2	1	62	2417.00
184	33	1	3	57	2553.00
185	12	2	2	52	1473.00
186	40	2	3	61	1424.00
187	33	1	3	62	2752.00
188	34	1	3	61	1977.00
189	32	1	3	56	2637.00
190	43	2	4	62	1537.00
191	44	2	4	61	1557.00
192	41	1	3	72	1967.00
193	33	1	3	57	1753.00
194	33	1	3	57	1654.00
195	37	2	3	64	1424.00
196	37	2	3	64	1354.00
197	54	1	3	47	1440.00
198	45	2	3	69	1155.00
199	41	1	3	61	1527.00
200	40	1	3	53	1435.00

BESTAND 41

1	0	0	1
1	1	0	1
1	1	0	1
1	1	1	1
1	1	0	0
1	0	1	1
1	1	1	0
1	0	0	1
1	1	1	0
0	1	1	1

BESTAND 42

1	1	1	1
1	0	0	1
1	0	1	1
1	0	0	1
1	1	1	1
0	1	1	1
1	0	1	1
0	0	1	1
1	0	0	1
0	0	1	1

BESTAND 43

1	1	1	0
1	1	1	1
0	1	1	1
1	0	0	0
1	1	1	1
1	1	0	1
0	1	0	1
0	1	0	0
1	1	1	1
1	0	0	1

BESTAND 44

1	0	0	1
1	1	0	1
1	1	0	0
1	0	0	1
1	0	0	1
1	1	0	0
0	0	1	1
1	0	1	1
1	0	0	1
1	1	1	1

BESTAND 45

1	1	1	0
1	0	0	1
1	1	0	1
1	0	1	1
1	1	1	1
0	1	1	1
1	1	0	1
1	0	1	1
0	0	0	1
1	1	0	1

BESTAND 46

1	1	1	1
1	1	0	1
1	1	1	1
1	0	0	1
1	1	0	0
1	1	1	0
1	0	1	1
0	0	0	1
0	1	1	0
1	1	1	1

BESTAND 47

0	0	0	1
1	0	1	1
1	0	1	1
1	1	1	1
1	1	0	1
1	0	1	1
0	1	0	1
1	1	0	0
0	0	0	1
1	0	0	1

BESTAND 48

1	1	0	0
1	1	1	1
0	0	1	1
0	0	1	0
1	0	0	1
1	0	1	1
1	0	1	0
1	1	0	0
1	1	0	1
1	0	0	1

BESTAND 49

1	1	1	1
1	0	1	1
1	1	1	1
1	0	0	1
1	1	0	1
1	0	0	1
1	0	1	0
0	1	0	1
1	1	1	1
1	1	1	1

BESTAND 50

0	0	1	1
1	1	0	1
0	1	0	0
0	0	0	1
1	0	1	1
1	0	0	1
0	1	1	1
1	1	1	1
0	0	1	1
1	0	0	1

BESTAND 51

WEG
DIK
FAT
MIS
KOP
OOI
ERG
EER
BEL
ZON
EGO
VAT
NEP
VAN
DUN
DOM
DEK
MUI
PEP
ZEE
WAT
OOR
DAM
ALT
DEN
POP
ZOT
VAL
MAN
LIP
VET
LOT
ENT
CEL
GAS
EED
UIT
MAF
GAT
DOG
AAI
PAP
BOA
EEN
ZUS
BIG
EIS
WEE
LAP
DOK
AAN
PIT

BEK
LOS
DOE
PIL
ZAK
LES
OUD
REL
DAT
RIT
DIA
DOL
HEL
SLA
ZES
BUI
SIP
DOT
KIP
SPA
DOR
BOT
KAT
ZET
BOF
JUK
WEI
BOL
BAS
IEP
TAS
ABT
HUT
UIL
AKS
TOP
ARM
TOT
WOL
BAR
BED
LUW
ELK
VEL
OOR
BIT
ION
URN

BESTAND 52

HIJ HAD ZICH NET ALS ADVOCaat GEVESTIGD, EN OP ZIJN KANTOOR DRAAIDE HIJ DOELLOOS IN ZIJN BUREAUSTOEL HEEN EN WEER.

PLOTSELING HOORDE HIJ STAPPEN OP DE GANG.

ALSOF HIJ HET HEEL DRUK HAD, NAM HIJ DE TELEFOON OP EN SPRAK: "NATUURLIJK MEVROUW, DEZE ZAAK WIL IK GAARNE VOOR U BEHANDELEN..."

NA EEN BESCHEIDEN KLOPJE WAS DE MAN BINNENGEKOMEN, EN IN EEN AFWACHTENDE HOUDING BIJ DE DEUR BLIJVEN STAAN. "...NEE MEVROUW, DAT KOST U... PARDON? ALS HET NIET CONVENIEERT, KUNT U GERUST NAAR EEN CONFRATER GAAN... ALS U LIEVER HEBT DAT IK HET DOE... GOED MEVROUW STORT U DAN ALVAST 275 GULDEN, DAN REGEL IK DE ZAAK VOOR U..."

MET EEN BLASE GEBAAAR LEGDE HIJ DE TELEFOON NEER EN WENDE ZICH TOT DE MAN DIE NOG STEEDS BESCHEIDEN BIJ DE DEUR STOND.

"WAT KAN IK VOOR U DOEN?"

DE MAN ZETTE ZIJN TAS OP DE GROND EN ZEIJ: "ALS HET U SCHIKT, WOU IK DE TELEFOON EVEN AANSLUITEN..."

MAX TAILLEUR

In bestand 52 komen geen hoofdletters voor.

Bestand 54 is ontstaan uit bestand 52 door

"verschuiving over" 17. Het eerste stukje

staat hieronder.

BESTAND 54

H YZA YPU QZTY EVK ROJ RUMETRRK XV

BESTAND 53

VERSCHIJN	PEFREIN	LURVEN	JAT	COURTISANE
IN	LEF	PEUK	SOUVEREIN	PENOSE
KONIJN	GOZEP	BELABBERD	GEIN	KIT
MOLLEN	SCHAVUIT	MIETERS	KEET	FIJN
BALEIN	KLEUNEN	OMKOPEN	SCHURK	KRAAK
HEITJE	CHANTAGE	LINK	POEN	TRONIE
MAJEM	KASSIAN	EVERZWIJN	KASTELEIN	MAZZEL
ROES	VERDWIJN	TREIN	PLEIN	LIJN
HOMMELES	CHAGRIJN	GRIEBUS	KOTER	OPROER
SCHOREM	DRSCUUR	KLEIN	GABBER	BAK
FILIGREIN	KEDIN	MIJN	MARSEPEIN	MUITERIJ
GIS	BINK	GORDIJN	PETOET	STUK
PIEREMENT	JAJEM	CRIMINEEL	JOEEL	MAFKIKKER
RIJKSHOTELMEP		DRIJFSIJS	KAPSONES	TRAMMELANT
PIMPELEN	BREIN	GREIN	SCHOFT	NEUTJE
BAJES	FIELT	ASTRANT	KEUTEL	SATIJN
SCHOBBEJAKREGIJN		SCHOELJE	KLOFFIE	BONJE
SJOECHE	BRANIE	DRIVE IN	RUS	KAMELENRUG
DOL	PONUM	DOKKEN	LUMMEL	CHARLATAN
TERREIN	KUL	ZWIJN	TEUT	WIJN

De onderstreepte letters zijn hoofdletters.

BESTAND 56

TOEN DE ROOKWOLKEN WAREN OPGETROKKEN, KON MEN DE GELEERDE PRWYTKOFSKY KLAGEND BIJ DE OVERBLIJFSELEN ZIEN ZITTEN.

"PRAW", ZO RIEP PRWYTKOFSKY UIT. "DAAR IS MIJNE GANSE KOSTELIJKE BUKKE IN DE LUCHT GESPRINGT. HET IS KOMMERVOL."

"BENT U IEMAND?" VROEG EEN STEM NAAST HEM.

DE HOOGLERAAR PRWYTKOFSKY KEEK GETROFFEN OP EN ZAG IN HET MOERAS EEN KLEINE GESTALTE STAAN.

"PRAW" SPRAK PRWYTKOFSKY, ZICH HERSTELLEND, "IK BEN DER DOCTOR PRWYTKOFSKY; HOOGLEAAR TE ROMMELDAM; EEN MAN DER WETENSCHAP."

"GOED," HERNAM HET VREEMDE VENTJE, "ZEG DAN TEGEN UW STAD-GENOTEN, DAT DE DROON WIL ONDERHANDELEN."

"ONDERHANDELEN?" HERHAALDE PROFESSOR PRWYTKOFSKY VERRAST.

"DAT IS JA SCHOON. VOEP MIJ NAAP HEM TOE."

"NEE," ZEI DE VREEMDELING, "UW KUKEL IS GEMETEN; HET WAS EEN MINKUKEL."

EEN OGENBLIK WAS PRWYTKOFSKY VERSLAGEN.

"PRAW" MOMPTELDE HIJ. "WELK EEN VERSCHRIKKELIJKE ONVER-SCHAAMDHEID... MAAR IK WEET WANNEER IK NIET ERWENST BEN..."

EN PRWYTKOFSKY STAPTE HEEN.

BESTAND 59

```

START
AANT:=0
1 A:=LEES
  ALSA=00DAN3
  ALSA=10DAN ANDERS1
2 AANT:=AANT+1
  NAAR
3 SCHRIJF:=AANT
  KLAAR
5+1+1+6+7+11+1+10+4+1+0
  
```

BESTAND 60

```

START
TEKST:="JAN FICTIEF"
GROOTSTE:=LEES
1 A:=LEES
  ALSA=00DAN4ANDERS2
2 ALSA>GROOTSTEDAN3ANDERS1
3 GROOTSTE:=A
  NAAR1
4 TEKST:="HET GROOTS?E GETAL
  NR
  TEKST:="UIT DEZE RIJ IS
  SCHRIJF:=GROOTSTE
  KLAAR
5+12+3+14 +9+0
  
```

BESTAND 61

```

START
SOM:=0
1 A:=LEES
  ALSA<1000DAN2ANDERS3
2 SOM:=SOM+A
  NAAR1
2 SCHRIJF:=SOM
  KLAAR
5+7+3-10+4+8-47+53+42-73+1000
  
```

BESTAND 62

```

START
SOMPOS:=0
SOMNEG:=0
1 A:=LEES
  ALSA<1000DAN2ANDERS4
2 ALSA<0DAN3
  SOMPOS:=SOMPOS+A
  NAAR1
  SOMNEG:=SOMNEG+A
  NAAR1
4 SCHRIJF:=SOMPOS
  SCHRIJF:=SOMNEG
  KLAAR
+4+5-3+7-3+4-10+13+23+1000
  
```

Hoofdletterwoorden zijn onderstreept.
Mogelijk zullen later in de program-
mateksten nog spaties worden ingelast.

BESTAND 63

```

START
1 A:=LEESSYM
  ALSA:=99DAN2
  ALSA:=86DAN3ANDERS4
3 TEKST:="ARCHIBALD"
4 SCHRIJFSYM:=A
  NAAR1
2 KLAAR
DE GROTE > ZEI TOT DE KLEINE
>, > IS EEN MOOIE NAAM.
  
```

BESTAND 68

AAMDEELD	AANSLAG	AGENDA	AMRIVALENTARABESK	AUTOMOBIEL
PANKPILJET	BEJEGENING	BESCHIMPEN	BLAFFER	BOEZEM
CONTINU	DAMSLAPER	DOCENT	DOPING	DUIKELAAR
GELAG	GENOT	GEVLIJ	HEETHOOFD	HETERDAAD
JUWEL	KANNIBAAL	KIESKEURTGLA	LEEP	LOVER
MAGNEET	MAST	MENHIR	NAVRANT	NOODLOT
OVERAL	PAARD	PARAFRASE	PEGEL	PIETLUT
POND	RUNNER	SCHERTS	SCHORT	SKI
STOFFER	STRALEN	SURLIEM	TRAP	TREND
U	JITSCHOT	VALS	VREEMD	WAFFEL
WIELRENNER	XYLOFOON	ZENUWEN		

BESTAND 64

250

639

558

890

-374

-281

48

238

633

-446

972

671

209

-302

924

-334

679

-75

-170

-312

-103

-75

869

949

969

126

510

-201

611

637

673

219

-383

-180

-475

967

967

-16

-115

329

53

707

770

505

28

780

559

-386

-217

-221

-224

-173

-450

755

510

129

128

564

952

697

235

802

60

911

983

382

344

-14

982

-272

-213

225

64

91

-122

909

-366

-498

-221

805

-348

336

613

290

-272

10

381

119

-147

803

494

-200

748

-198

-55

107

843

968

-405

711

374

-160

83

438

616

844

872

151

-270

-248

412

757

240

236

421

845

-365

78

808

-64

98

983

212

-128

-146

25

657

-179

648

218

-146

-89

-153

-288

-399

-13

166

111

334

994

-480

921

-213

-14

0

-380

276

-244

-26

351

623

95

815

-255

-314

678

-246

863

-474

568

351

-228

699

349

103

233

168

886

165

-471

839

206

-293

900

-270

506

385

129

-242

-212

-53

219

151

-387

459

228

-8

485

-347

207

-122

-135

-93

-229

211

292

345

795

613

650

566

738

466

550

992

581

571

-166

-152

403

-57

707

259

573

-278

503

481

-424

709

559

252

447

540

-273

766

355

965

-495

-223

789

-234

-203

-177

881

294

294

-418

853

-361

-16

-157

582

411

779

112

804

-338

283

836

925

BESTAND 65

CLUB TAAL INTRODUCTIE PAGINA VOORAL MOGELIJKHEID GEDAANTEVERMISSELING LIJN. REFREIN
OP ORGANISATIE U VERENIGING BEHENDIGHEIDSPROEVEN PARAGRAF MIJN TELEFOONNUMMER ZMIJN
UI ZADEL SPOORWEGMAATSCHAPPIJ OPENING EN LIJNMAATSCHAP VERZAMELING UIT IJDELTUIT AARDEI
ZMOEL BLOEDKORAAL OPEN PLEIN SPOORWEGEMPLACEMENT GEDAANTEVERANDERING TELEFOONBOEK
AF PA IJSSAAN KASTELEIN CITROEN UITGANG IJZERVILSEL HOEFLAG AARDEWERKFAIRIEK HARNACHEMENT
IJSSCHOTS WATERVERF QUOTUM NIEUWSGIERIGHEID FLEEST

Centrum Uithof